# **Develop Outlook Add-ins with SharePoint Framework**

Overview

SharePoint Framework is spreading its wings and is now ready to go beyond SharePoint and develop solutions targeting the Office 365 platform. SPFx already supports hosting the SPFx web part as tabs in MS Teams and create personal tabs in MS Team. Now, SPFx v1.10 allows to develop Outlook Web App add-ins hosted on SharePoint.

In this article, we will explore how can develop Outlook Add-ins with SharePoint Framework.

**Develop SPFx Web Part for Outlook Add-ins**

Before we dive into implementing SPFx web part, please note that this feature is available in beta and at the moment, only supported within the context of the Outlook Web Access. The GA should support Office desktop too. As this feature is in preview, we have to scaffold the project with Yeoman using *--plusbeta* switch.

Follow the below steps to develop the SPFx solution,

**Step 1**

Open a command prompt. Create a directory for the SPFx solution.

1. md spfx-outlook-addin

**Step 2**

Navigate to the directory.

1. cd spfx-outlook-addin

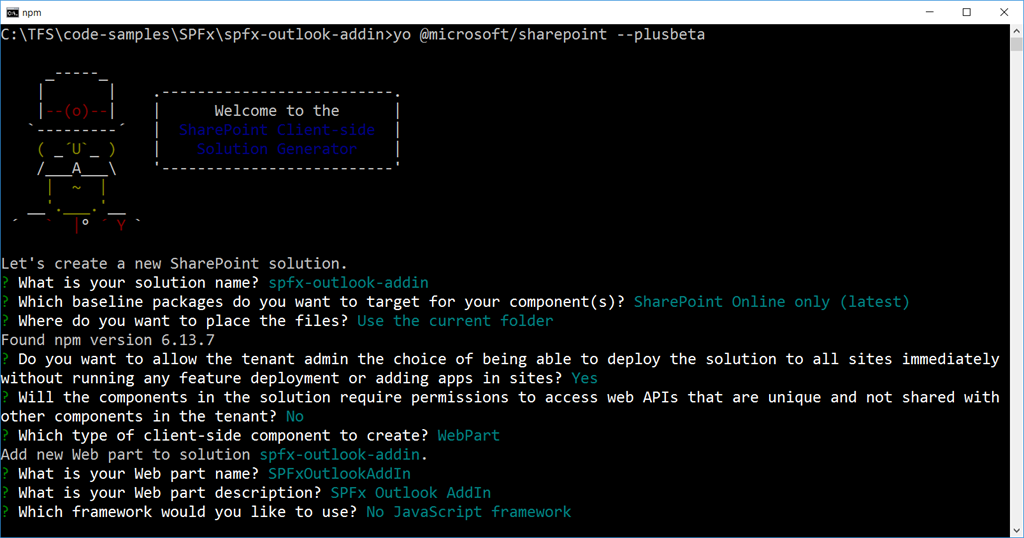
**Step 3**

Run the Yeoman SharePoint Generator to create the solution.

1. yo @microsoft/sharepoint --plusbeta

**Step 4**

Yeoman generator will present you with the wizard by asking questions about the solution to be created.



When prompted:

* Accept the default **spfx-outlook-addin** as your solution name, and then select Enter.
* Select **SharePoint Online only (latest)**, and then select Enter.
* Select **Use the current folder** as the location for the files.
* Select **y** to ensure that your web part is automatically deployed tenant-wide when it's added to the tenant App Catalog.
* Select **N** on the question if solution contains unique permissions.
* Select **WebPart** as the client-side component type to be created.
* Enter **SPFxOutlookAddIn** for the web part name, and then select Enter.
* Enter the description of the web part.
* Select **No JavaScript framework** to develop the web part.

**Step 5**

Yeoman generator will perform the scaffolding process to generate the solution. The scaffolding process will take a significant amount of time.

**Step 6**

In the command prompt type the below command to open the solution in the code editor of your choice.

1. code .

Use Office JavaScript SDK (Office.js)

Include correct types by installing below npm package. The type declarations should go in devDependencies.

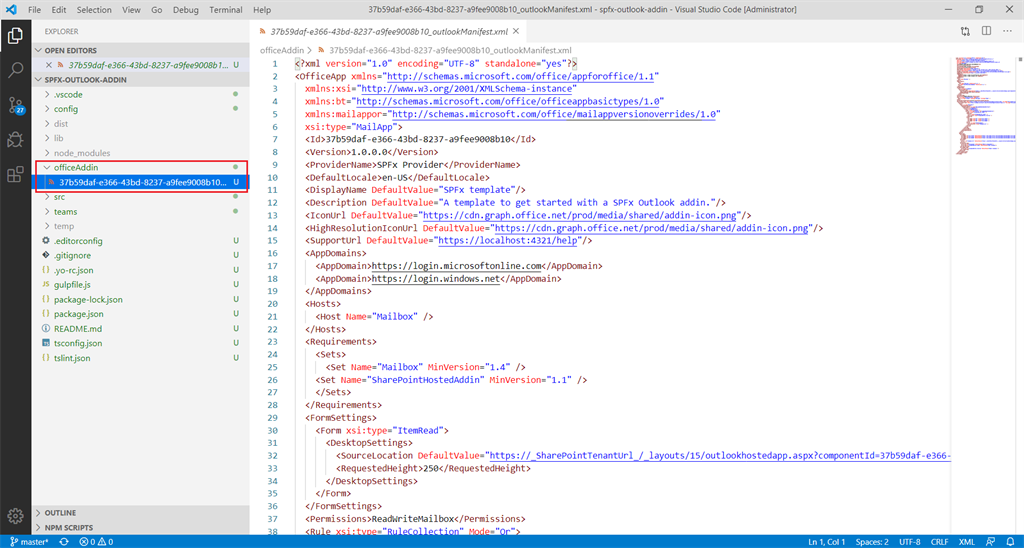
1. npm install @types/office-js --save-dev

The below property will help to identify if the web part is running under Office context.

1. this.context.sdks.office

**Support for Office AddIn**

The SPFx solution contains an additional folder “officeAddin”. This supports a onetime configuration option when the add-in is initially viewed. This option is controlled from URL of the solution.



By default, configuration option is enabled. If your add-in does not have any initial configuration options, you can remove the isConfigureMode query parameter.

**Code the SPFx Outlook AddIn**

We will update the code to work as Outlook AddIn.

**Step 1**

Open the web part file at “src\webparts\spFxOutlookAddIn\SpFxOutlookAddInWebPart.ts”

**Step 2**

Update the render method to check the office context.

1. **public** render(): **void** {
2. let title: **string** = "";
3. let subTitle: **string** = "";
4. let contextInfo: **string** = "";
6. **if** (**this**.context.sdks.office) {
7. // Office context
8. title = "Welcome to Office!";
9. subTitle = "Extending Office with SPFx.";
10. contextInfo = "Email: " + **this**.context.sdks.office.context.mailbox.userProfile.emailAddress;
11. }
12. **else** {
13. // SharePoint context
14. title = "Welcome to SharePoint!";
15. subTitle = "Customize SharePoint experiences using Web Parts.";
16. contextInfo = "SharePoint site: " + **this**.context.pageContext.web.title;
17. }
19. **this**.domElement.innerHTML = `
20. <div **class**="${ styles.spFxOutlookAddIn}">
21. <div **class**="${ styles.container}">
22. <div **class**="${ styles.row}">
23. <div **class**="${ styles.column}">
24. <span **class**="${ styles.title}">${title}</span>
25. <p **class**="${ styles.subTitle}">${subTitle}</p>
26. <p **class**="${ styles.description}">${contextInfo}</p>
27. </div>
28. </div>
29. </div>
30. </div>`;
31. }

**Deploy Add-in**

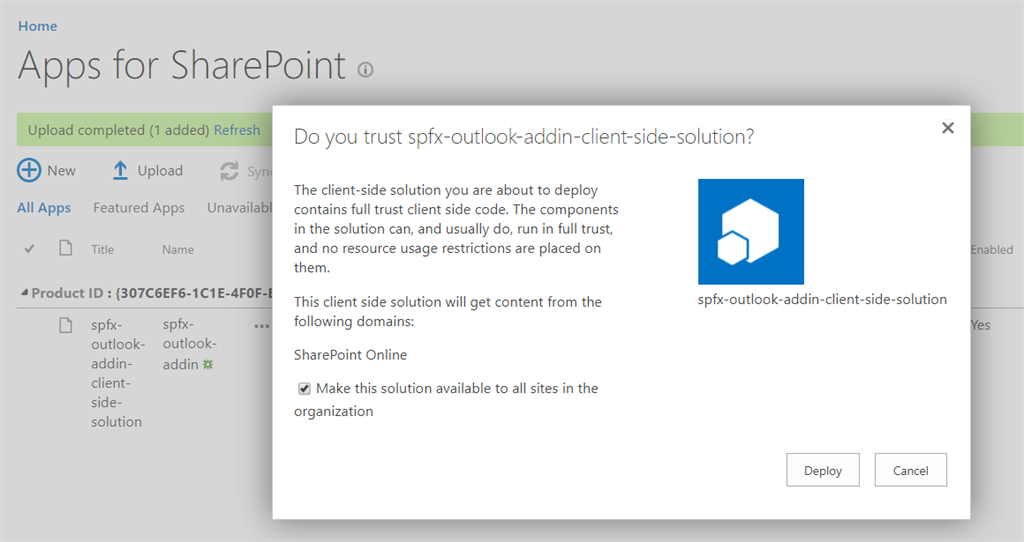
The deployment process involves 2 steps.

**Step 1 - Deploy solution to SharePoint App Catalog**

Package and deploy the solution by running the below commands:

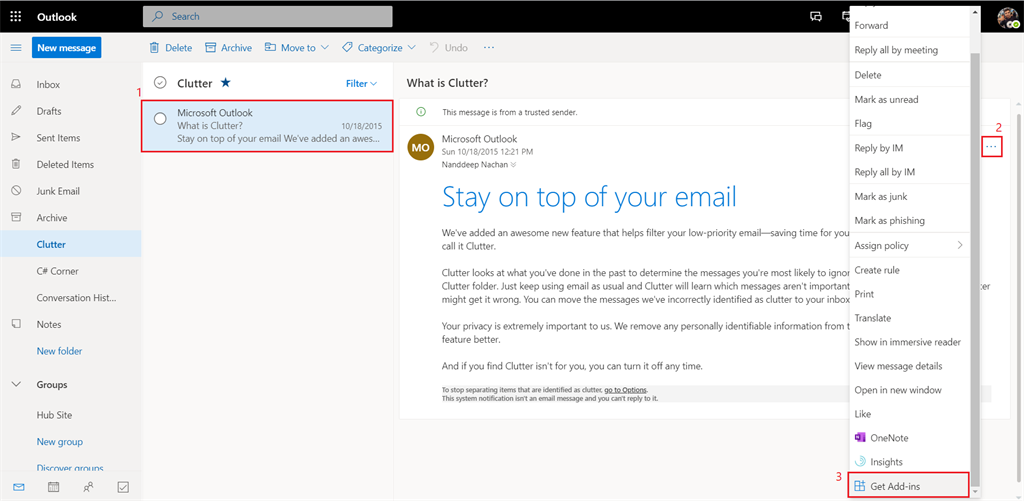
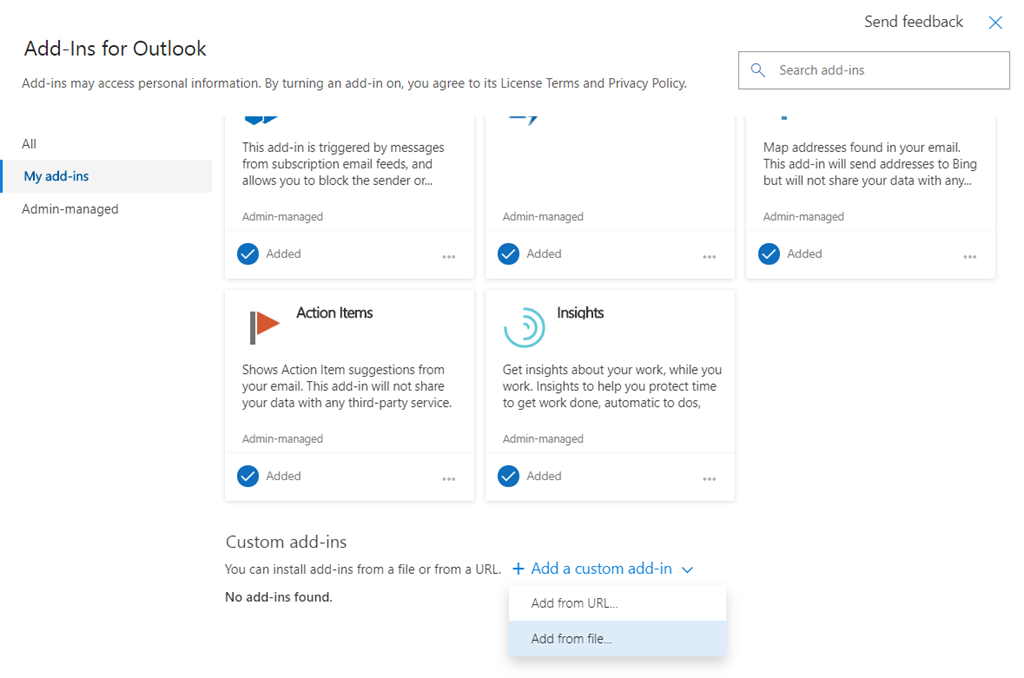
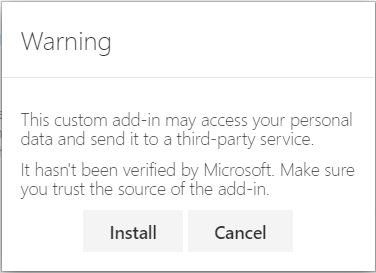
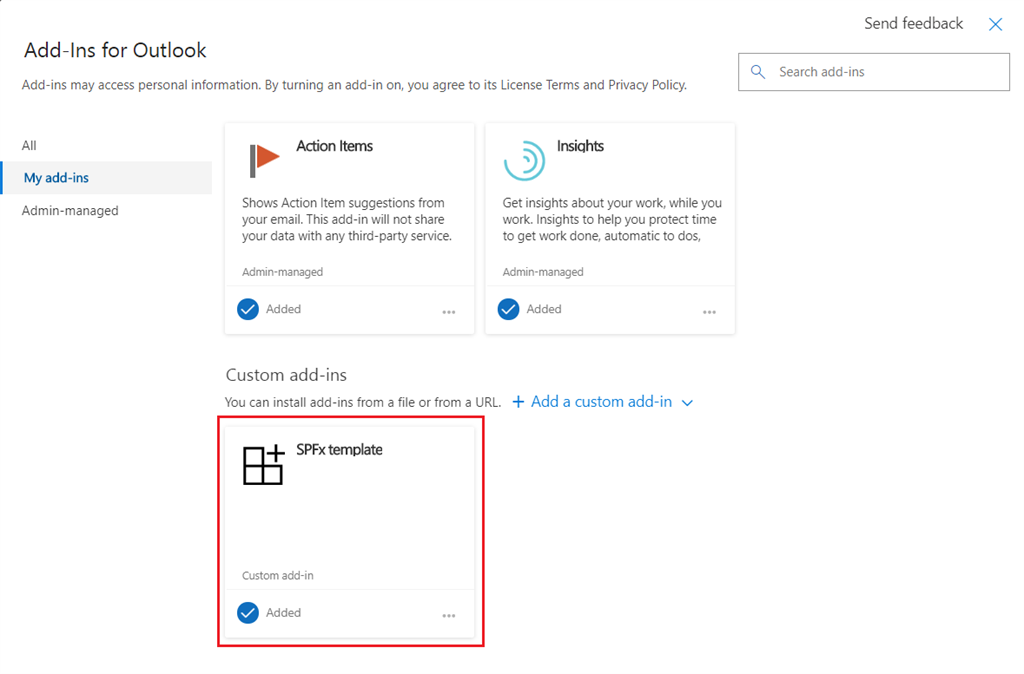
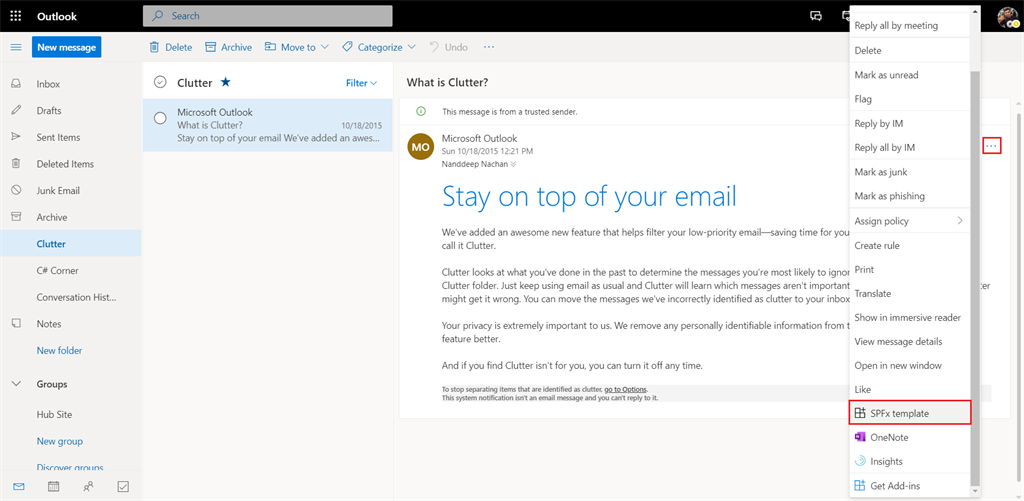
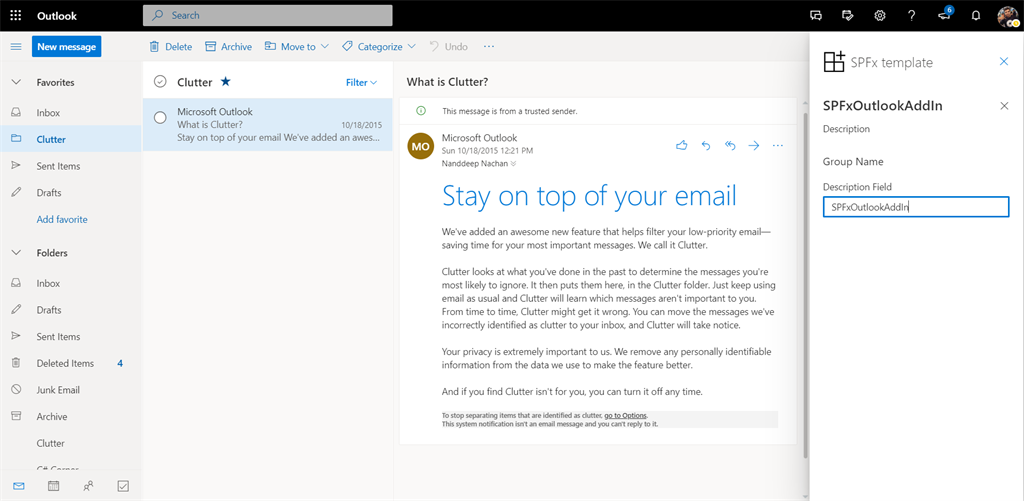
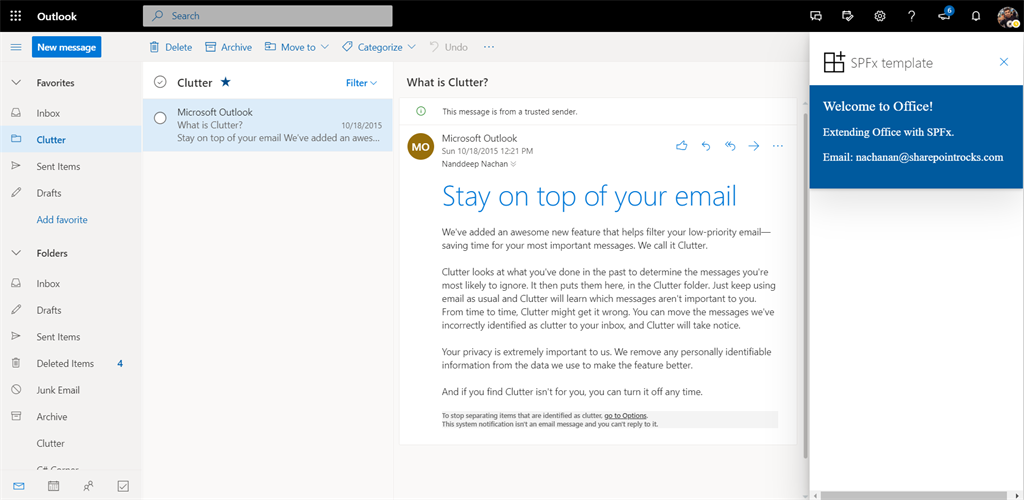
1. gulp bundle --ship
2. gulp package-solution --ship

Deploy the solution to SharePoint app Catalog.



**Step 2 - Add Add-in to Outlook**

Since this feature is in preview, it can only be added to Outlook web access.

1. Open Office 365 Outlook Web access.
2. Open any existing email.
3. Click “…” and Select “Get Add-ins”.  
     
   
4. From left menu, click “My add-ins”.
5. Under “Custom add-ins”, click “Add from file…”  
   
6. Upload the manifest xml file from SPFx project solution under the officeAddin folder.
7. Click Install.  
     
   
8. The custom add-in will appear.  
   
9. Close the add-in window.
10. From an email, click “…”  
      
    
11. The configuration option is available during initial view of the add-in.  
      
    
12. After the initial configuration options are set, the SPFx Outlook AddIn can be seen as below,  
      
    

**Summary**

SPFx v1.10 allows us to develop Outlook Web App add-ins hosted on SharePoint. This feature is available in beta and now, only supported within the context of the Outlook Web Access.