

TEAMS APPS

HANDS-ON LAB

Using the Teams Platform to surface a Talent Management App in Teams

ABSTRACT

This lab document contains a step-by-step guide to create a Teams App that allowed a recruitment team to manage the positions and candidates all within Teams.

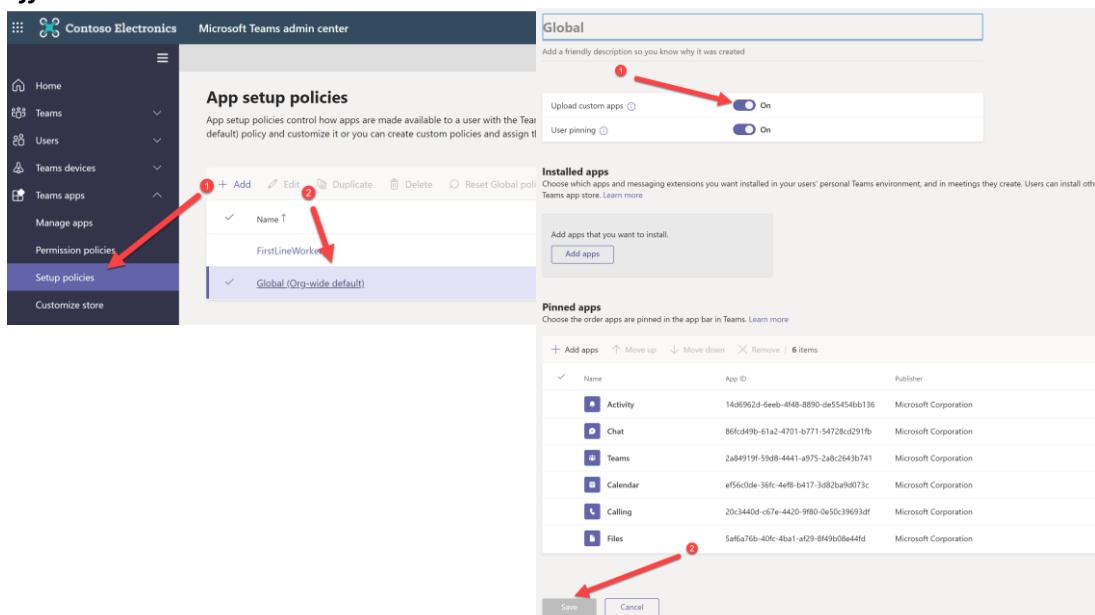
About

Written for the App Modernisation Series – a series of events for Microsoft Partners created by Jason Cabot, Mike Ormerod, Scott Perham, Luciana Blanchard and Jack Lewis. Credit to the original team behind this Teams app hands on lab, found here - <https://github.com/OfficeDev/msteams-sample-contoso-hr-talent-app>

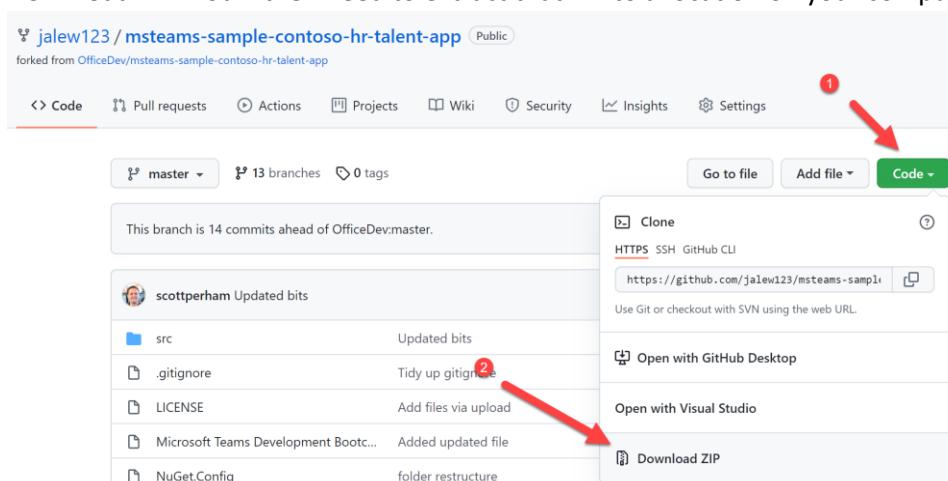
Getting Started

You will need the following to complete this lab:

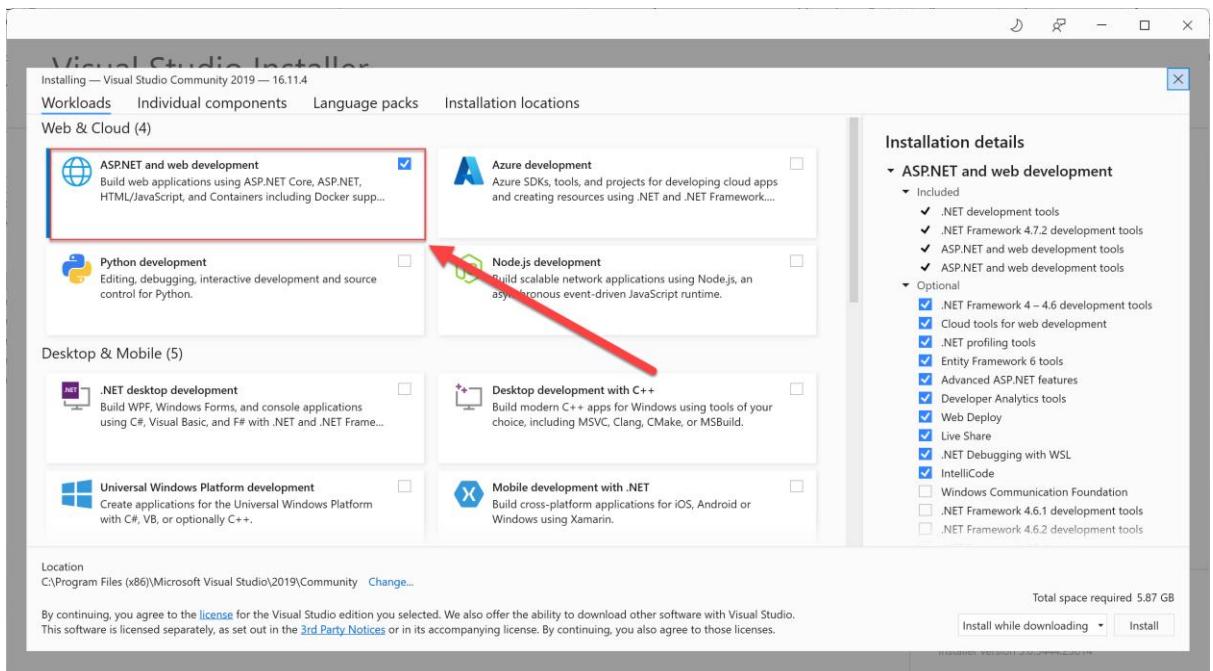
1. **A Microsoft 365 E5 Tenant**, which can be obtained from <https://cdx.transform.microsoft.com/> (for Microsoft Partners) or by signing up to the Microsoft 365 Developer Program – [Welcome to the Microsoft 365 Developer Program | Microsoft Docs](#). Alternatively, a Microsoft 365 E5 Trial will also provide the functionality you need. We do **not** recommend using your production/live Tenant for this lab. We will refer to this Tenant as the **Test/Dev Tenant** in this documentation.
2. **Enable Side-Loading in Teams**. To do this, go to <https://admin.teams.microsoft.com/> and login as an administrator in your Test/Dev Tenant. Click **Setup policies** and select the **Global (Org-wide default)** policy. Ensure **Upload Custom Apps** is **On**, click **Save** if you have set this from Off to On. **Note: if this is not enabled, it may take up to 8 hours for this policy to take effect.**



3. **The Talent Management App**, which can be downloaded from [here](#), click **Code** and select **Download ZIP**. You'll then need to extract that ZIP to a location on your computer.



4. **Visual Studio**, installed onto your local machine. If you do not have Visual Studio, the Community edition can be downloaded [here](#). When installing Visual Studio, make sure to select **ASP.NET and web development**.



5. **.Net 5 SDK**, which is required to run the Talent Management App, available [here](#).
6. **NGROK**, a tunnelling tool, which will allow you to run your app within Teams, available [here](#).
You'll need to extract the NGROK ZIP to a location on your computer.
7. **Postman**, a tool that allows you to send HTTP requests from your computer, available [here](#).
8. **Azure Subscription**, required to create the Azure Bot Service instance that will be used within your Teams app. This resource does not cost anything so if you want to use an existing subscription that is fine. Alternatively, sign up for an Azure trial [here](#).

Talent Management App Lab

Overview

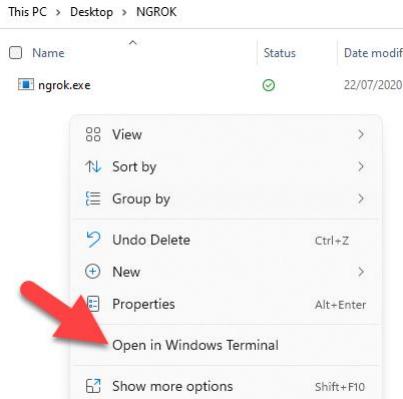
During this lab, you will learn how to build and deploy a Microsoft Teams app in Office 365 that will be used by the Human Resources department within their Microsoft Teams clients. The app will facilitate the department's hiring of new talent into the organization, provide immediate interview feedback, schedule interview loops, and improve the overall hiring process of new employees. The lab is divided into several exercises that will help you understand how to transform hiring and candidate management flow of new talent and make it more interactive and responsive for HR teams and interviewees.

Lab Steps

- 1) Setup Talent Management App

In this step we will setup the NGROK Tunnel and open and configure the Talent Management App in Visual Studio. Setup the Azure AD App Registration & Bot Service Resource in Azure. And then run (debug) the Talent Management App locally, within Visual Studio.

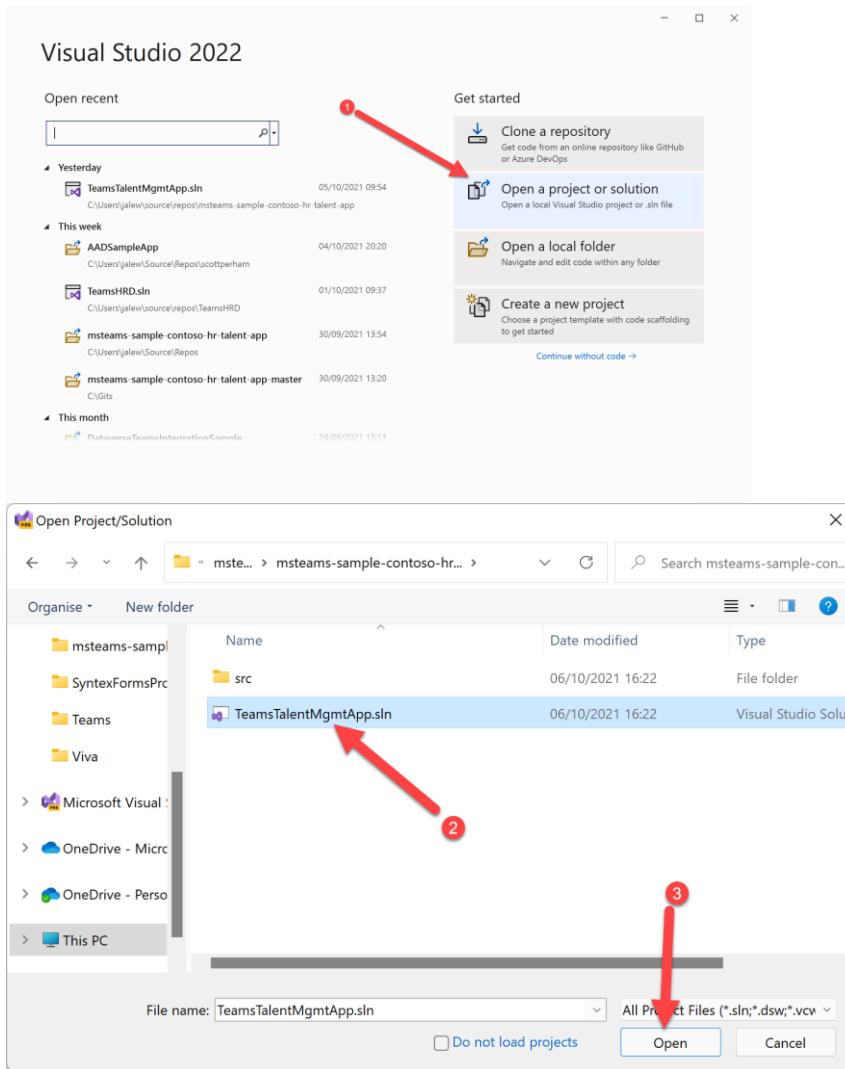
1. Open the folder where you extracted the NGROK ZIP to, and start a Command Prompt/Windows Terminal session here.



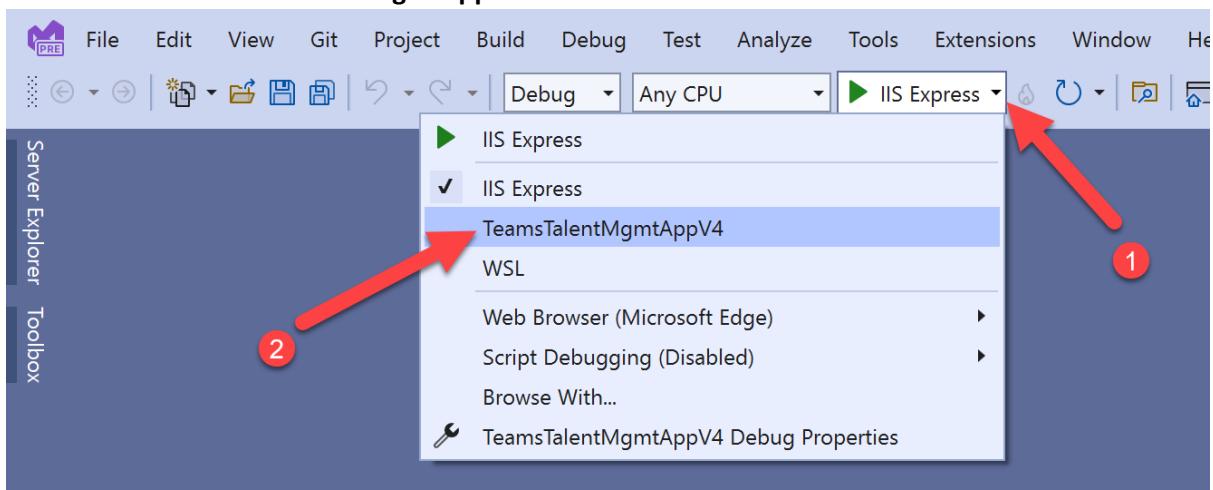
- Enter the following in the Command Prompt/Windows Terminal session: `.\ngrok http 5000`
This will open an NGROK tunnel and forward traffic destined for the forwarding URLs to localhost:5000 (where you will run the Talent Management app later in this lab).
Make note of the HTTPS forwarding URL. You will need this later when you configure the Teams app. Do not close the NGROK window. Make sure this window is left open.

```
Windows PowerShell - ngrok.exe
ngrok by @inconshreveable
Session Status          online
Account
Region                Europe (eu)
Version               2.3.35
Web Interface          http://127.0.0.1:4040
Forwarding             http://cbe47bb9c336.eu.ngrok.io -> http://localhost:5000
Forwarding             https://cbe47bb9c336.eu.ngrok.io -> http://localhost:5000
Connections            ttl     opn      rt1     rt5     p50     p98
                        0       0       0.00    0.00    0.00    0.00
```

- Open Visual Studio and open the **TeamsTalentMgmtApp.sln** file found within the extracted Talent Management Repo you downloaded from GitHub.



- When Visual Studio has loaded the solution and associated projects, we need to set the debug properties correctly. In the Visual Studio toolbar, select the **IIS Express dropdown menu** and select **TeamsTalentMgmtAppV4**.



- On the right-side of Visual Studio, within Solution Explorer, expand **src**, **TeamsTalentMgmtAppV4**, **Properties** and select **launchSettings.json**. This file contains the properties that are used when we run (debug) the Talent Management App locally, in Visual Studio. Change the value of **launchBrowser** (line 20) to **false**. Review the **applicationUrl**

values (line 24) – these should use port 5001 and 5000 (for HTTPS and HTTP respectively).

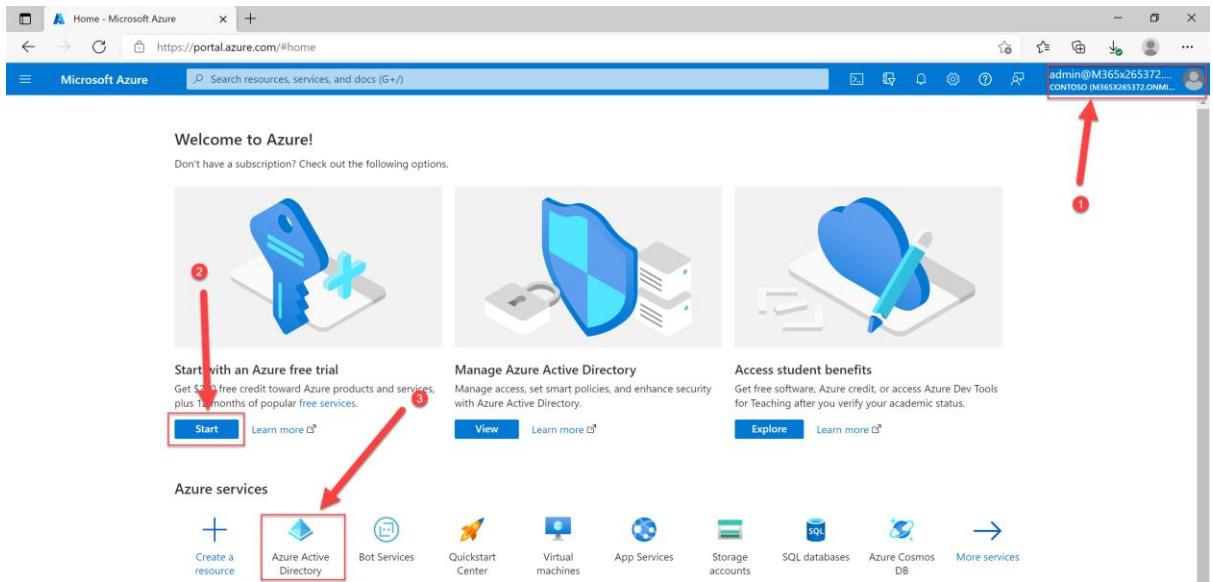
Hit **CTRL+S** to save or click the Save icon in the toolbar.

```
launchSettings.json
Schema: https://json.schemastore.org/launchsettings.json
1  {
2    "iisSettings": {
3      "windowsAuthentication": false,
4      "anonymousAuthentication": true,
5      "iisExpress": {
6        "applicationUrl": "http://localhost:51593/",
7        "sslPort": 44319
8      }
9    },
10   "profiles": {
11     "IIS Express": {
12       "commandName": "IISExpress",
13       "launchBrowser": true,
14       "environmentVariables": {
15         "ASPNETCORE_ENVIRONMENT": "Development"
16       }
17     },
18     "TeamsTalentMgmtAppV4": {
19       "commandName": "Project",
20       "launchBrowser": false,
21       "environmentVariables": {
22         "ASPNETCORE_ENVIRONMENT": "Development"
23       },
24       "applicationUrl": "https://localhost:5001;http://localhost:5000"
25     }
26   }
27 }
```

6. Select **appSettings.json**. This file contains placeholder values for **BaseUrl**, **TeamsAppId**, **MicrosoftAppId**, **MicrosoftDirectoryId**, **MicrosoftAppPassword**, **ApplicationIdUri** & **OAuthConnectionName**. Before we can run this app, we need to replace these with actual values. Currently, we only know the **BaseUrl** value - it is the NGROK URL. Replace the value for **BaseUrl** (line 2) with the **NGROK HTTPS URL Value** (gathered during step 2 of this section of the lab). Hit **CTRL+S** to save or click the Save icon in the toolbar.

```
appSettings.json
Schema: https://json.schemastore.org/appsettings.json
1  {
2    "BaseUrl": "https://jalem123.eu.ngrok.io/",
3    "TeamsAppId": "PASTE_YOUR_TEAMS_APP_ID_HERE",
4    "MicrosoftAppId": "PASTE_YOUR_BOT_APPID_HERE",
5    "MicrosoftDirectoryId": "PASTE_YOUR_DIRECTORY_ID_HERE",
6    "MicrosoftAppPassword": "PASTE_YOUR_BOT_APP_PASSWORD_HERE",
7    "ApplicationIdUri": "PASTE_YOUR_APP_URI_HERE",
8    "OAuthConnectionName": "PASTE_YOUR_CONNECTION_NAME",
9    "Logging": {
10      "LogLevel": {
11        "Default": "Warning"
12      }
13    },
14    "AllowedHosts": "*"
15 }
```

7. We now need to setup the Azure AD App Registration and Azure Bot Service resource in Azure. This will provide us with the remaining values for the appSettings.json file. Navigate to <https://portal.azure.com/> and sign in with the Administrator account, that has access to an Azure Subscription. If you do not have an Azure Subscription, sign in with your Test/Dev Tenant Administrator, and sign up for an Azure Free Trial. Once you are signed in and have access to an Azure Subscription on that account, Select **Azure Active Directory**.



8. Select App Registrations and then click New Registration

Home > Contoso

Contoso | App registrations

Azure Active Directory

New registration

The new App registrations search preview! Click!

2 June 30th, 2020 we will no longer add any new applications to Microsoft Authentication Library (MSAL) or

All applications Owned applications Deleted

Start typing a name or Application ID to filter these

Display name

- CO ContosoTeamsCreationApp
- SA SAMLAPP
- TE TeamsSAMLApp
- and

1

9. Enter a name for your App, such as **Talent-Management-App**, select **Accounts in any organizational directory (Any Azure AD directory – Multitenant)**, select **Single-page application (SPA)** from the dropdown menu and then enter your Redirect URI as **NGROKURL/StaticViews/LoginResult.html** – click **Register**.

Home > Contoso >
Register an application ...

* Name

The user-facing display name for this application (this can be changed later).

Talent-Management-App

1

Supported account types

Who can use this application or access this API?

Accounts in this organizational directory only (Contoso only - Single tenant)

Accounts in any organizational directory (Any Azure AD directory - Multitenant) 2

Accounts in any organizational directory (Any Azure AD directory - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)

Personal Microsoft accounts only

[Help me choose...](#)

Redirect URI (optional)

We'll return the authentication response to this URI after successfully authenticating the user. Providing this now is optional and it can be changed later, but a value is required for most authentication scenarios.

Single-page application (SPA) https://jalew123.eu.ngrok.io/StaticViews/LoginResult.html

3

4

Register an app you're working on here. Integrate gallery apps and other apps from outside your organization by adding from [Enterprise applicatic](#)

By proceeding, you agree to the [Microsoft Platform Policies](#)

Register

5

10. On the Overview page, copy the **Application (client) ID** and enter this as the **MicrosoftAppId** value in `appsettings.json` (line 4).

Display name	: Talent-Management-App	Copy to clipboard
Application (client) ID	: 5652156c-2020-42c4-b6f2-b942a45c526e	
Object ID	: 4f390244-f81b-45bf-93ff-f2f2d2bbb5f9	
Directory (tenant) ID	: d06ea5c6-1047-45d0-8ea9-7de7d40e3c58	
Supported account types	: Multiple organizations	

```

appsettings.json
Schema: https://json.schemastore.org/appsettings.json
1  {
2    "BaseUrl": "https://jalew123.eu.ngrok.io/",
3    "TeamsAppId": "████████████████████████████████",
4    "MicrosoftAppId": "5652156c-2020-42c4-b6f2-b942a45c526e", █
5    "MicrosoftAppDirectoryID": "████████████████████████████████",
6    "MicrosoftAppPassword": "████████████████████████████████",
7    "ApplicationIdUri": "████████████████████████████████",
8    "OAuthConnectionName": "████████████████████████████████"
  }

```

11. Copy the Directory (tenant) ID and enter this as the **MicrosoftDirectoryID** value in `appsettings.json` (line 5).

The screenshot shows the Azure portal interface for managing app registrations. On the left, there's a sidebar with options like Overview, Quickstart, Integration assistant, Manage, Branding, Authentication, Certificates & secrets, Token configuration, API permissions, and several JSON files (appsettings.json, manifest.json*, Startup.cs, common.js, Login.html, launchSettings.json). The main area displays the app's details under the 'Essentials' section. The 'Display name' is 'Talent-Management-App'. The 'Application (client) ID' is '5652156c-2020-42c4-b6f2-b942a45c526e'. The 'Object ID' is '4f390244-f81b-45bf-93ff-f2f2d2b'. The 'Directory (tenant) ID' is 'd06ea5c6-1047-45d0-8ea9-7de7d40e3c58'. A red box highlights the 'Directory (tenant) ID' field, and a red arrow points to the 'Copy to clipboard' button next to it. Below these fields, it says 'Supported account types: Multiple organizations'. There are also two informational messages at the bottom.

12. Select **Authentication**, click **Add a platform**, select **Web**, enter the Redirect URI of <https://token.botframework.com/auth/web/redirect> and click **Configure**. For context, these Redirect URIs are the allowed locations that Azure AD will redirect clients to when an access token has been received via a successful authentication. We need to configure the Bot Framework URI, as Bot SSO uses this address when obtaining authorisation/consent. The SPA URI is used to redirect browsers back to your LoginRedirect.html page, which is the page used to process the Azure AD access token, and then store it in Local Storage for use when calling the Talent Management APIs. Feel free to inspect the LoginRedirect.html & Login.html pages to see what they do, and how to use the latest MSAL Browser libraries to enable SSO support for this app.

Configure Web

Platform configurations

Depending on the platform or device this application is targeting, additional configuration may be required.

Single-page application

Redirect URIs

The URIs we will accept as destinations when returning authentication responses (tokens) after successfully authenticating or signing out users. Also referred to as reply URLs. [Learn more about Redirect URIs and their restrictions](#)

Grant types

Your Redirect URI is eligible for the Authorization Code Flow with PKCE.

Front-channel logout URL

This is where we send a request to have the application clear the user's session data. This is required for single sign-out to work correctly.

Implicit grant and hybrid flows

Request a token directly from the authorization endpoint. If the application has a single-page architecture, doesn't use the authorization code flow, or if it invokes a web API via JavaScript, select both access tokens and ID tokens. For ASP.NET Core web apps and other web apps that use hybrid authentication, select only ID tokens. [Learn more](#)

Configure

13. Select Certificates & secrets, select New client secret and click Add

Add a client secret

Description: Enter a description for this client secret

Expires: 12 months

Description	Expires	Value	Secret ID
No client secrets have been created for this application.			

Add

14. Once generated, copy the Value of the Client Secret and enter this as the MicrosoftAppPassword value in appsettings.json (line 6).

Client secrets

A secret string that the application uses to prove its identity when requesting a token. Also can be referred to as application password.

+ New client secret

Description	Expires	Value	Copy to clipboard	Secret ID
Password uploaded on Thu Oct 07 2021	10/7/2022	Jj.7Q~YeYOr-R2q118u1cKR2GanYhkY_41B...		ab5671d6-7958-4265-b4ba-1a24da64f04a

```

appsettings.json manifest.json* Startup.cs common.js Login.html launchSettings.json
Schema: https://json.schemastore.org/appsettings.json

1 "BaseUrl": "https://jalew123.eu.ngrok.io/",
2 "TeamsAppId": "REDACTED",
3 "MicrosoftAppId": "5652156c-2020-42c4-b6f2-b942a45c526e",
4 "MicrosoftAppDirectoryID": "d06ea5c6-1047-45d0-8ea9-7de7d40e3c58",
5 "MicrosoftAppPassword": "Jj.7Q~YeYOr-R2q118u1cKR2GanYhKY_41Bd1",
6 "ApplicationIdUri": "REDACTED",
7 "OAuthConnectionName": "REDACTED"
8

```

15. Select API permission and click Add a permission.

Home > Contoso > Talent-Management-App

Talent-Management-App | API permissions

Search (Ctrl+ /) Refresh Got feedback?

Overview Quickstart Integration assistant

Manage

- Branding
- Authentication
- Certificates & secrets
- Token configuration
- API permissions** (highlighted with a red box)
- Expose an API
- App roles
- Owners
- Roles and administrators | Preview
- Manifest

Starting November 9th, 2020 end users will no longer be able to grant consent to newly registered multitenant apps without verified publishers. Add MPN ID to verify publisher

The "Admin consent required" column shows the default value for an organization. However, user consent can be customized per permission, user, or app. This column may not reflect the value

Configured permissions

Applications are authorized to call APIs when they are granted permissions by users/admins as part of the consent process. The list of configured permissions should include all the permissions the application needs. Learn more about permissions and consent

API / Permissions name	Type	Description	Admin consent requ...	Status
User.Read	Delegated	Sign in and read user profile	No	...

To view and manage permissions and user consent, try Enterprise applications.

16. Select Microsoft Graph Request API permissions

Select an API

Microsoft APIs APIs my organization uses My APIs

Commonly used Microsoft APIs

Microsoft Graph

Take advantage of the tremendous amount of data in Office 365, Enterprise Mobility + Security, and Windows 10. Access Azure AD, Excel, Intune, Outlook/Exchange, OneDrive, OneNote, SharePoint, Planner, and more through a single endpoint.

Azure Rights Management Services Allow validated users to read and write protected content	Azure Service Management Programmatic access to much of the functionality available through the Azure portal	Data Export Service for Microsoft Dynamics 365 Export data from Microsoft Dynamics CRM organization to an external destination
Dynamics 365 Business Central Programmatic access to data and functionality in Dynamics 365 Business Central	Dynamics CRM Access the capabilities of CRM business software and ERP systems	Flow Service Embed flow templates and manage flows

17. Select Delegated permissions, select openid, profile and then search for and select TeamsAppsInstallation.ReadForUser. Click Add permissions.

Request API permissions

What type of permissions does your application require?

1

Delegated permissions
Your application needs to access the API as the signed-in user.

Application permissions
Your application runs as a background service or daemon without a signed-in user.

Select permissions **2**

TeamsAppInstallation.ReadForUser

The "Admin consent required" column shows the default value for an organization. However, user consent can be customized per permission, user, or app. This column may not reflect the value in your organization, or in organizations where this app will be used. [Learn more](#)

Permission	Admin consent required
TeamsAppInstallation (1)	
<input checked="" type="checkbox"/> TeamsAppInstallation.ReadForUser ⓘ Read user's installed Teams apps	No

3

4

Add permissions Discard

18. If successful, your API Permissions should look like the below:

Home > Contoso > Talent-Management-App

Talent-Management-App | API permissions

Search (Ctrl+ /) Refresh Got feedback?

Overview Quickstart Integration assistant

⚠️ You are editing permission(s) to your application, users will have to consent even if they've already done so previously.

⚠️ Starting November 9th, 2020 end users will no longer be able to grant consent to newly registered multitenant apps without verified publishers. Add MPN ID to verify publisher

The "Admin consent required" column shows the default value for an organization. However, user consent can be customized per permission, user, or app. This column may not reflect the value in your organization, or in organizations where this app will be used.

Manage

- Branding
- Authentication
- Certificates & secrets
- Token configuration
- API permissions** (4)
- Expose an API
- App roles
- Owners
- Roles and administrators | Preview
- Manifest

Support + Troubleshooting

Troubleshooting New support request

Configured permissions

Applications are authorized to call APIs when they are granted permissions by users/admins as part of the consent process. The list of configured permissions should include all the permissions the application needs. [Learn more about permissions and consent](#)

+ Add a permission Grant admin consent for Contoso

API / Permissions name	Type	Description	Admin consent requ...	Status
openid	Delegated	Sign users in	No	...
profile	Delegated	View users' basic profile	No	...
TeamsAppInstallation.ReadForUser	Delegated	Read user's installed Teams apps	No	...
User.Read	Delegated	Sign in and read user profile	No	...

19. Select **Expose an API** and Set the Application ID URI.

Home > Contoso > Talent-Management-App

Talent-Management-App | Expose an API

Search (Ctrl+ /) < Got feedback?

Overview Quickstart Integration assistant

Manage

- Branding
- Authentication
- Certificates & secrets
- Token configuration
- API permissions
- Expose an API** (highlighted)
- App roles
- Owners
- Roles and administrators | Preview
- Manifest

Scopes defined by this API

Define custom scopes to restrict access to data and functionality protected by the API. An application that requires access to parts of this API can request that a user or admin consent to one or more of these.

Adding a scope here creates only delegated permissions. If you are looking to create application-only scopes, use 'App roles' and define app roles assignable to application type. [Go to App roles](#).

+ Add a scope

Scopes	Who can consent	Admin consent display ...	User consent display na...	State
No scopes have been defined				

Authorized client applications

Authorizing a client application indicates that this API trusts the application and users should not be asked to consent when the client calls this API.

+ Add a client application

Client Id	Scopes
No client applications have been authorized	

20. The Application ID URI needs to be configured in the following format:

api://NGROKURL/botid-MICROSOFTAPPID – click Save.

Set the App ID URI

Application ID URI

api://jalew123.eu.ngrok.io/botid-5652156c-2020-42c4-b6f2-b942a45c526e

Save **Discard**

21. Once saved, copy the Application ID URI and enter this as the **ApplicationIdUri** value in **appsettings.json** (line 7).

Home > TalentManagementBot > Talent-Management-App

Talent-Management-App | Expose an API

Search (Ctrl+ /) < Got feedback? Copy to clipboard

Overview Quickstart Integration assistant

Manage

- Branding
- Authentication
- Certificates & secrets
- appsettings.json** (highlighted)
- manifest.json*
- Startup.cs
- common.js
- Login.html
- launchSettings.json

Schema: <https://json.schemastore.org/appsettings.json>

```

1  {
2    "BaseUrl": "https://jalew123.eu.ngrok.io/",
3    "TeamsAppId": "████████████████████████████████████████",
4    "MicrosoftAppId": "5652156c-2020-42c4-b6f2-b942a45c526e",
5    "MicrosoftAppDirectoryID": "d06ea5c6-1047-45d0-8ea9-7de7d40e3c58",
6    "MicrosoftAppPassword": "Jj.7Q~YeY0r-R2q118u1cKR2GanYhkY_41Bd1",
7    "ApplicationIdUri": "api://jalew123.eu.ngrok.io/botid-5652156c-2020-42c4-b6f2-b942a45c526e",
8    "OAuthConnectionName": "████████████████████████████████████████"
  
```

22. Select **Add a scope**

23. We now need to configure the API scope that will be used when SSO into your application has been completed both inside and outside of Microsoft Teams. Enter the Scope name **access_as_user** select **Admins and users**. For the display names and descriptions enter **access_as_user** (note: in a real world scenario you would populate these with more information). Select **Add scope**

Add a scope ×

1 Scope name * ✓

2 Who can consent? Admins and users Admins only

3 Admin consent display name * ✓

4 Admin consent description *

5 User consent display name ✓

6 User consent description

State Enabled Disabled

7 Add scope Cancel

24. We now need to enable the Teams Native/Mobile and Web applications to swap their access tokens for an access token for the Talent Management Application. Select **Add a client application**.

25. Teams is identified by 2 separate Client IDs, so we need to repeat this step twice. Enter **1fec8e78-bce4-4aaaf-ab1b-5451cc387264** as the Client ID, select the **access_as_user** scope and click **Add Application**. Then click **Add a client application** again, enter **5e3ce6c0-2b1f-4285-8d4b-75ee78787346** as the Client ID, select the **access_as_user** scope and click **Add Application**.

You have now authorised the Teams Apps to swap their access tokens for an access token for your application. More on this later in the lab!

26. The final step we need to do, is to configure the App Registration manifest to only use Access Tokens v2. Select **Manifest**, change the value of **accessTokenAcceptedVersion** to **2** (line 4). Select **Save**.

The screenshot shows the Microsoft Azure portal interface for managing an app registration. On the left, there's a sidebar with various management options like Overview, Quickstart, Integration assistant, and API permissions. Under the 'Manage' section, the 'Manifest' option is highlighted with a red box and labeled '1'. In the main content area, there's a search bar, a toolbar with Save, Discard, Upload, Download, and Got feedback? buttons, and a note about updating the application via its JSON representation. The JSON code itself is displayed, with line 4 being the key one: "accessTokenAcceptedVersion": 2. A red box highlights this line, and a red arrow labeled '2' points to it. A third red arrow labeled '3' points to the 'Save' button at the top right.

```

1 {
2   "id": "4f390244-f81b-45bf-93ff-f2f2d2bbb5f9",
3   "acceptMappedClaims": null,
4   "accessTokenAcceptedVersion": 2, // Red box highlights this line
5   "addIns": [],
6   "allowPublicClient": null,
7   "appId": "5652156c-2020-42c4-b6f2-b942a45c526e",
8   "appRoles": [],
9   "oauth2AllowUrlPathMatching": false,
10  "createdDateTime": "2021-10-07T11:29:03Z",
11  "certification": null,
12  "disabledByMicrosoftStatus": null,
13  "groupMembershipClaims": null,
14  "identifierUris": [
15    "api://jalew123.eu.ngrok.io/botid-5652156c-2020-42c4-b6f2-b942a45c526e"
16  ],
17  "informationalUrls": {
18    "termsOfService": null,
19    "support": null,
20    "privacy": null,
21    "marketing": null
22  },
23  "keyCredentials": [],
24  "knownClientApplications": [],
25  "logoUrl": null,
26  "logoutUrl": null,
27  ...
28 }
  
```

27. The Azure AD App Registration has now successfully been configured. We now need to configure the Azure Bot Services resource, that is linked to the Azure AD App Registration you have just created. In the Search bar at the top of the Azure Portal, search for and select **Applied AI services**.

The screenshot shows the Microsoft Azure portal search results. The search bar at the top contains the text 'applied ai'. Below the search bar, a list of services is shown, with 'Applied AI services' being the first item and highlighted with a red box and labeled '2'. Other items in the list include Cognitive Search, Form recognizers, Immersive readers, Metrics advisors, Bonsai, Mailjet Email Service, Maintenance Configurations, Availability sets, and Test Emails. To the right of the search results, there are sections for Marketplace, Documentation, and Resource Groups.

28. Find Bot Services and click **Create**.

https://portal.azure.com/#blade/Microsoft_Azure_ProjectOxford/AppliedAIHub/overview

Applied AI services

Bot services

Metrics advisor

Video analyzer

Bot services

Cognitive search

Form recognizer

Immersive reader

Metrics advisor

Video analyzer

Bot services

+ Create

Metrics advisor

+ Create

Video analyzer

+ Create

Immersive reader

+ Create

29. Scroll down and select **Azure Bot** (note: you may need to select **load more** at the bottom of the page to see Azure Bot!)

Home > Applied AI services >

Bot Services

- AudioCodes
- Virtual Scheduling , ChatBot
- Speech to text
- Think AI Bot for Connectwise
- Vernacular.ai Intelligent Voice Assistant
- devNXT- AI driven smart application development
- Rx.Health
- Zammo.ai SaaS
- Audite Cloud
- Mia - Workplace Virtual Assistant
- Azure Health Bot
- Azure Bot

30. Select **Create**

The screenshot shows the Microsoft Azure portal interface. At the top, there's a blue header bar with the Microsoft Azure logo and a search bar. Below the header, the URL path is shown: Home > Applied AI services > Bot Services > Azure Bot. The main content area features a thumbnail of a blue cube icon labeled 'Azure Bot'. Below the icon, it says 'Microsoft' and has a rating of '★ 3.0 (5 Azure ratings)'. A prominent blue 'Create' button is centered below the icon, which is highlighted with a red rectangular box. Below the 'Create' button, there are tabs for 'Overview', 'Plans', 'Usage Information + Support', and 'Reviews'. The 'Overview' tab is underlined, indicating it is the active page.

31. At this point, you may see the following error page – this is because the user you are logged in as has no access to an Azure Subscription with funding/credits. If you see this, either sign-in as another user with access to an Azure Subscription or sign up for an Azure Free trial!

Home > Applied AI services > Bot Services > Azure Bot >
Switch directories ...

Switch directories

You are currently signed into the 'Contoso (M365x26537.onmicrosoft.com)' directory which does not have any subscriptions. You have other directories you can switch to or you can sign up for a new subscription.

Switch directories
Start free

If you see this, either sign in as a user with an Azure Subscription, or sign up for a Free Azure Trial Subscription

32. Assuming you have the Azure Subscription in place, you will see this interface. Enter a Bot handle name such as **TalentManagementBot** (note: these must be globally unique). Select an **Azure Subscription**, create or select an existing **Resource Group** for the Azure Bot to reside within. Select a **location** for the resource group if you are creating a new resource group. Change from Standard pricing to **Free**. For the Microsoft App ID, it's important that you select **Use an existing app registration**, enter your **Microsoft App ID** and **App Secret** that you generated earlier in this lab. Select **Review + Create**.

Home > Azure Bot >
Create an Azure Bot ...

Basics Tags Review + create

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Bot handle * ⓘ

1 TalentManagementBot ✓

Subscription * ⓘ

2 Azure subscription 1 ✓

Resource group * ⓘ

3 (New) TalentManagementBot ✓

Create new

New resource group location ⓘ

4 UK South ✓

Pricing

Select a pricing tier for your Azure Bot resource. You can change your selection later in the Azure portal's resource management. Learn more about available options, or request a pricing quote, by visiting the [Azure Bot Services pricing](#)

Pricing tier *

5 Free
Change plan

Microsoft App ID

A Microsoft AppID is required to create an Azure Bot resource. An App ID can be automatically created below, or you can manually create your own, then return here to input your new App ID and password. [Manually create App ID](#)

ⓘ The app secret will be stored in Azure Key Vault in the same resource group as your Azure bot. [Learn more](#) ⓘ

Microsoft App ID

Create new Microsoft App ID

Use existing app registration

Existing app id *

6 5652156c-2020-42c4-b6f2-b942a45c526e ✓

Existing app password *

7 ✓

Review + create

< Previous

Next : Tags >

33. Once Validation has passed, select **Create**.

Validation Passed

Basics Tags Review + create

Basics

Bot handle	TalentManagementBot
Subscription	Azure subscription 1
New resource group location	UK South
Pricing tier	Free
Microsoft App ID	Use existing app registration
Existing app id	5652156c-2020-42c4-b6f2-b942a45c526e
Existing app password	*****

Create < Previous Next Download a template for automation

34. Select **Go to resource** once the resource has been deployed

Home > CreateAzureBot_dx-20211007132314 | Overview

Deployment

Search (Ctrl+ /) < Delete Cancel Redeploy Refresh

Overview Deployment Inputs Outputs Template

Your deployment is complete

Deployment name: CreateAzureBot_dx-20211007132314
Subscription: Azure subscription 1
Resource group: TalentManagementBot

Start time: 10/7/2021, 1:30:11 PM
Correlation ID: 9ad61b69-7796-4d5f-90ce-76bd7ef0448b

Deployment details (Download)
Next steps

Go to resource

35. Select **Configuration**, enter the **Messaging endpoint** in the format of **NGROKURL/api/messages** – select **Apply** and then click **Add OAuth Connection Settings**.

Home > CreateAzureBot_dx-20211007132314 > TalentManagementBot

TalentManagementBot | Configuration

Azure Bot

1 Configuration

2 Messaging endpoint
https://jalew123.eu.ngrok.io/api/messages

Enable Streaming Endpoint

App Type
MultiTenant

Microsoft App ID (Manage) ⓘ
5652156c-2020-42c4-b6f2-b942a45c526e

Application Insights Instrumentation key ⓘ
Instrumentation key (Azure Application Insights key)

Application Insights API key ⓘ
API key (User-Generated Application Insights API key)

Application Insights Application ID ⓘ
Application ID (Application Insights Application ID)

Schema Transformation Version
V1.3

This determines how Bot Service converts messages sent between your bot and channels. [Learn more](#)

No OAuth Connection settings defined

3 Add OAuth Connection Settings

4 Apply Discard changes

36. Enter the name as **AAD**, select **Azure Active Directory v2** as the Service Provider, the Client ID is your **MicrosoftAppID**, the Client secret is your **MicrosoftAppPassword**, the Token Exchange URL if your **ApplicationIDURI**, Tenant ID is **Organizations** and Scopes is **user.read profile openid TeamsAppInstallation.ReadForUser** – click Save and Apply.

Messaging endpoint
https://jalew123.eu.ngrok.io/api/messages

Enable Streaming Endpoint

App Type
MultiTenant

Microsoft App ID (Manage) ⓘ
5652156c-2020-42c4-b6f2-b942a45c526e

Application Insights Instrumentation key ⓘ
Instrumentation key (Azure Application Insights key)

Application Insights API key ⓘ
API key (User-Generated Application Insights API key)

Application Insights Application ID ⓘ
Application ID (Application Insights Application ID)

Schema Transformation Version
V1.3

This determines how Bot Service converts messages sent between your bot and channels. [Learn more](#)

Name Service Provider Client id Client secret Token Exchange URL Tenant ID Scopes

Add OAuth Connection Settings

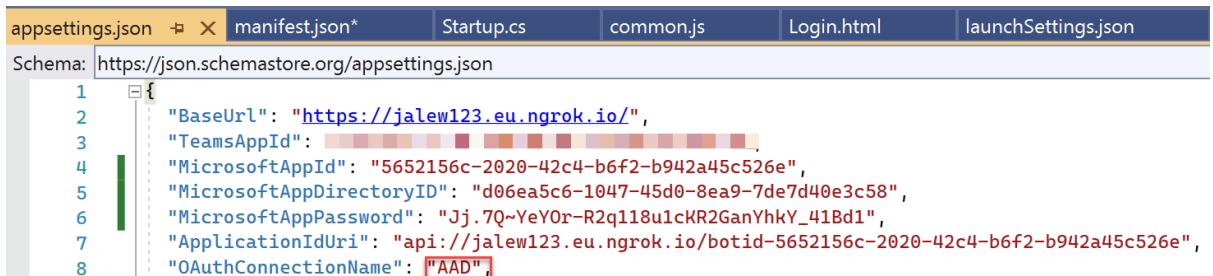
9 Apply Discard changes

New Connection Setting

1	Name *	AAD
2	Service Provider *	Azure Active Directory v2
3	Client id *	5652156c-2020-42c4-b6f2-b942a45c526e
4	Client secret *
5	Token Exchange URL	api://jalew123.eu.ngrok.io/botid-5652156c-2020-...
6	Tenant ID	Organizations
7	Scopes	user.read profile openid TeamsAppInstallation.Re...



37. Back into Visual Studio, set the **OAuthConnectionName** value in `appsettings.json` (line 8) to **AAD** – click CTRL+S to save the `appsettings.json` file.



```

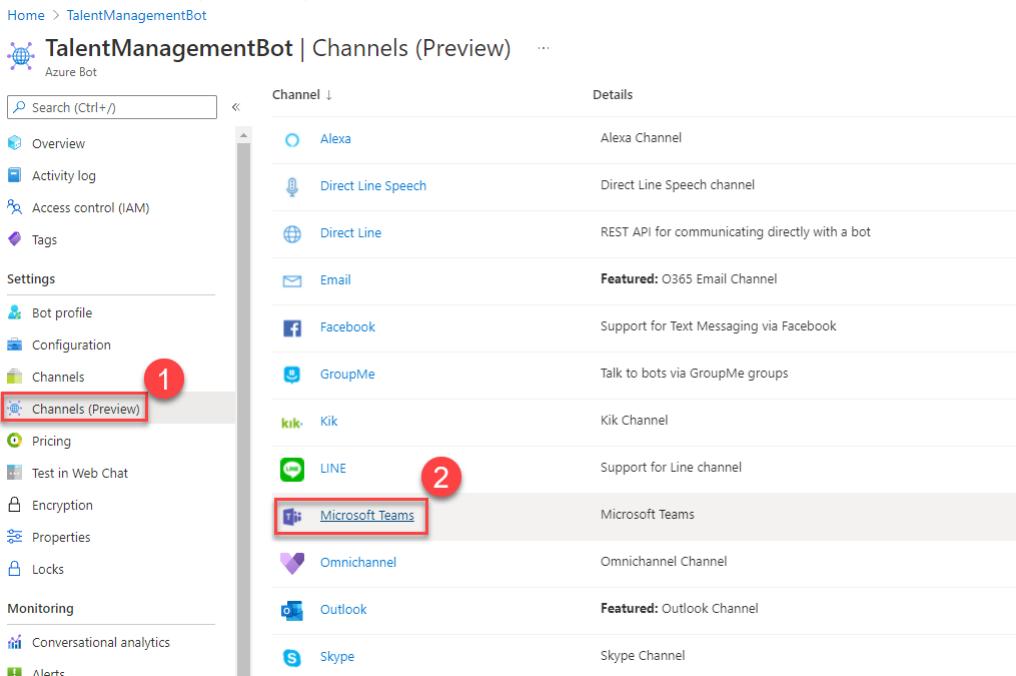
appsettings.json  manifest.json*  Startup.cs  common.js  Login.html  launchSettings.json
Schema: https://json.schemastore.org/appsettings.json

1
2   "BaseUrl": "https://jalew123.eu.ngrok.io/",
3   "TeamsAppId": "REDACTED",
4   "MicrosoftAppId": "5652156c-2020-42c4-b6f2-b942a45c526e",
5   "MicrosoftAppDirectoryID": "d06ea5c6-1047-45d0-8ea9-7de7d40e3c58",
6   "MicrosoftAppPassword": "Jj.7Q-YeYOr-R2q118u1cKR2GanYhkY_41Bd1",
7   "ApplicationIdUri": "api://jalew123.eu.ngrok.io/botid-5652156c-2020-42c4-b6f2-b942a45c526e",
8   "OAuthConnectionName": "AAD"

```

38. Now go back into the Azure Portal and configure the channels for the bot to support Teams.

Select **Channels (Preview)** and select **Microsoft Teams**.



Home > TalentManagementBot

TalentManagementBot | Channels (Preview)

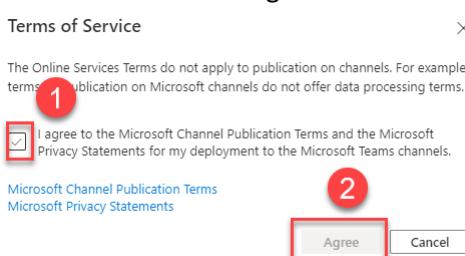
Azure Bot

Search (Ctrl+ /)

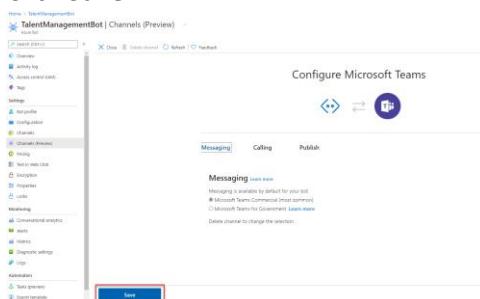
- Overview
- Activity log
- Access control (IAM)
- Tags
- Channels**
- Channels (Preview)** (highlighted with a red box)
- Pricing
- Test in Web Chat
- Encryption
- Properties
- Locks
- Monitoring
- Conversational analytics
- Alerts

Channel	Details
Alexa	Alexa Channel
Direct Line Speech	Direct Line Speech channel
Direct Line	REST API for communicating directly with a bot
Email	Featured: O365 Email Channel
Facebook	Support for Text Messaging via Facebook
GroupMe	Talk to bots via GroupMe groups
Kik	Kik Channel
LINE	Support for Line channel
Microsoft Teams (highlighted with a red box)	Microsoft Teams
Omnichannel	Omnichannel Channel
Outlook	Featured: Outlook Channel
Skype	Skype Channel

39. Tick the **checkbox** to agree to the terms of service and select **Agree**.



40. Click **Save**.



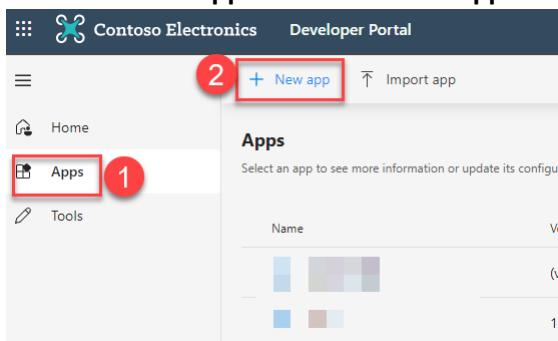
Configure Microsoft Teams

Messaging (highlighted with a red box)

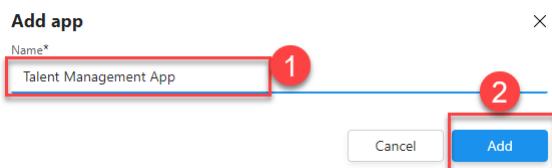
Save

41. Your Bot is now configured to forward messages received onto your App (via the NGROK URL) and is configured to work in Teams. The final step, before running the app locally, is to create a Teams App Manifest and populate the final value (TeamsAppID) in the appsettings.json file. To do this go to the **Teams Developer Portal**

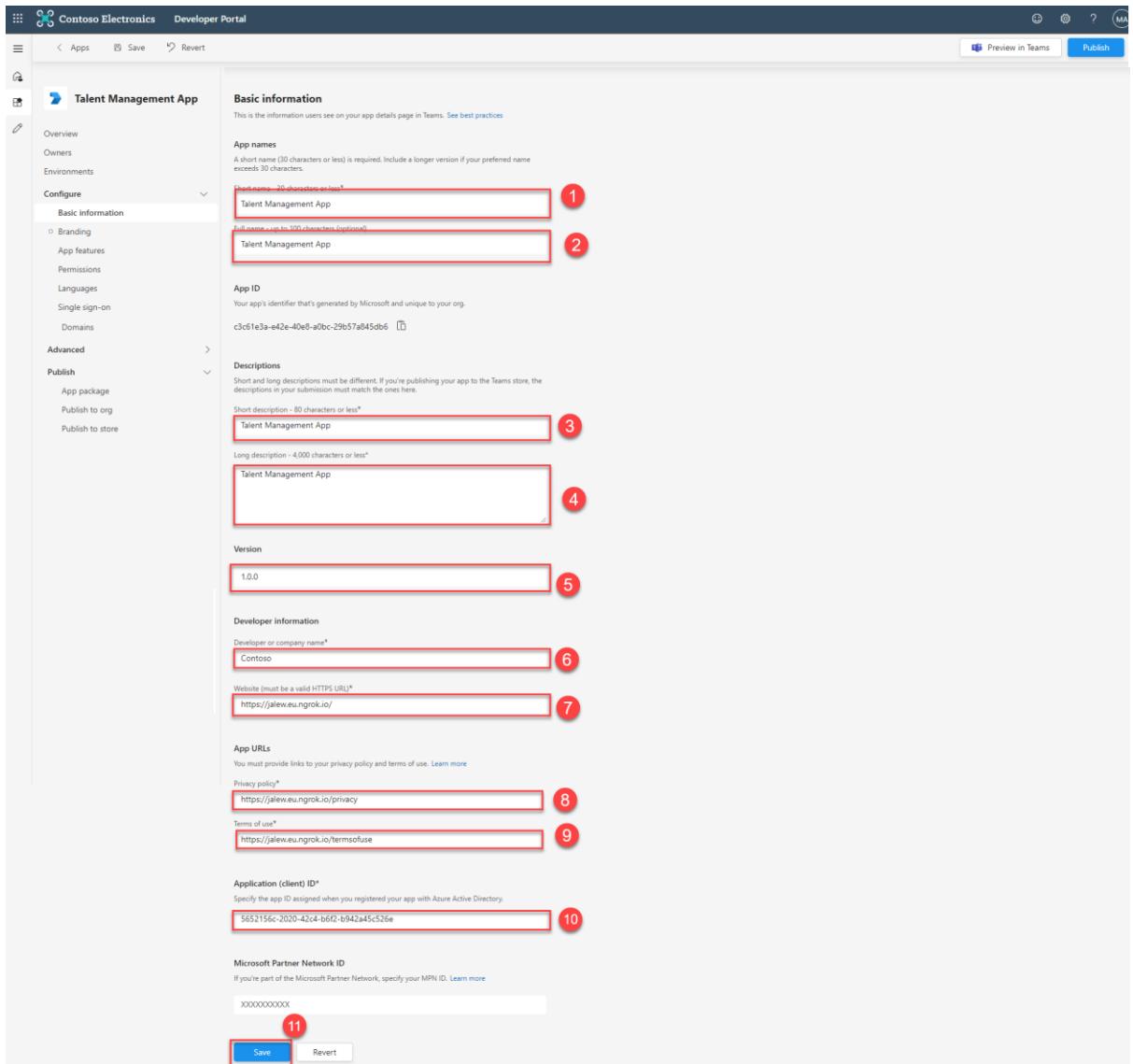
<https://dev.teams.microsoft.com/> and sign in with the Test/Dev Tenant Administrator account. Select **Apps** and click **+ New App**.



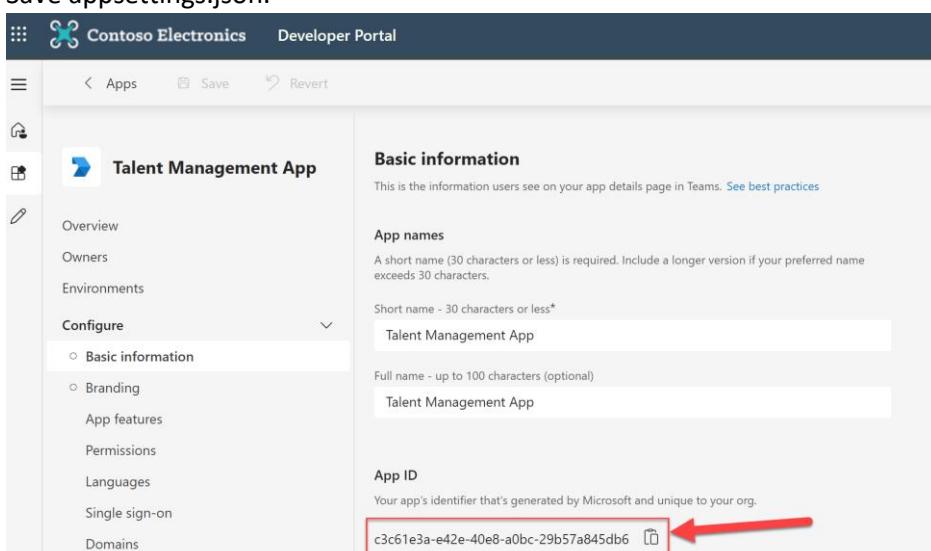
42. Enter the name **Talent Management App** and click **Add**.



43. We now need to enter some placeholder values into the Basic Information configuration of your Teams app. Start with copying the Short Name **Talent Management App**, and paste it into the **Full Name**, **Short Description** and **Long Description** fields. Set the version as **1.0.0**. Developer to **Contoso**. Website to your **NGROKURL**. Privacy Policy URL to **NGROKURL/privacy** and Terms of Use URL to **NGROK/termsofuse**. For Application (Client) ID, enter your **MicrosoftAppID**. Click **Save**.



44. Copy the **App ID** and paste this into appsettings.json as the value of **MicrosoftAppID** (line 3).
Save appsettings.json.

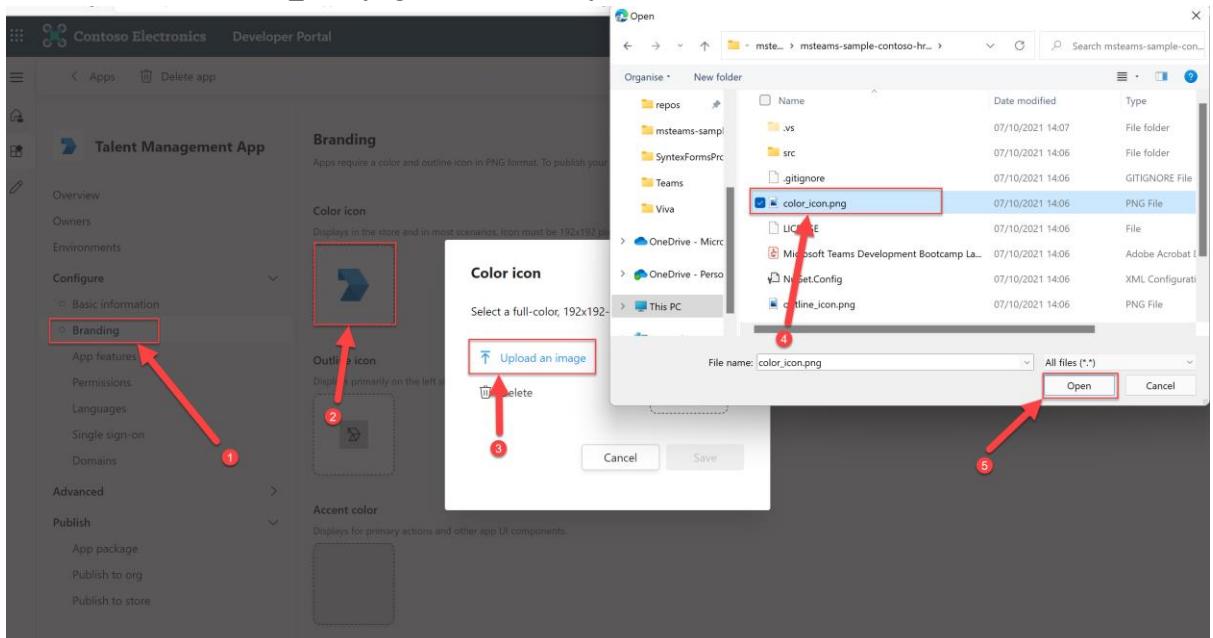


```

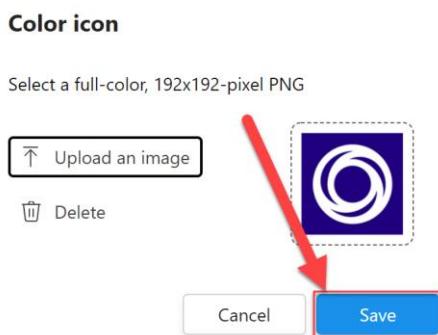
appsettings.json manifest.json* Startup.cs common.js Login.html launchSettings.json
Schema: https://json.schemastore.org/appsettings.json
1 "BaseUrl": "https://jalew123.eu.ngrok.io/",
2 "TeamsAppId": "c3c61e3a-e42e-40e8-a0bc-29b57a845db6",
3 "MicrosoftAppId": "5652156c-2020-42c4-b6f2-b942a45c526e",
4 "MicrosoftAppDirectoryID": "d06ea5c6-1047-45d0-8ea9-7de7d40e3c58",
5 "MicrosoftAppPassword": "Jj.7Q-YeYOr-R2q118u1CKR2GanYhkY_41Bd1",
6 "ApplicationIdUri": "api://jalew123.eu.ngrok.io/botid-5652156c-2020-42c4-b6f2-b942a45c526e",
7 "OAuthConnectionName": "AAD",
8

```

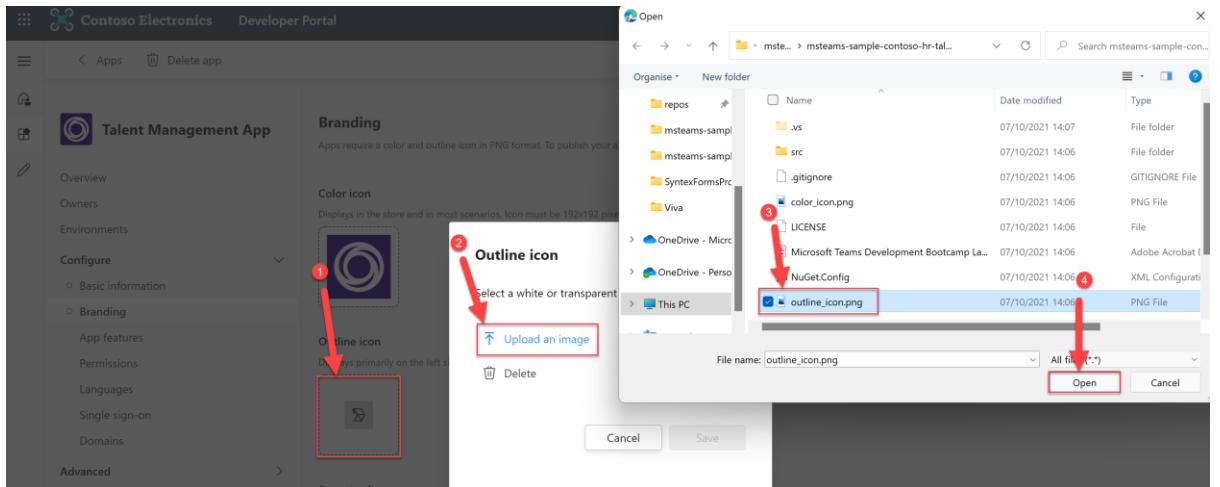
45. Go back into the Teams Developer Portal, and select **Branding**, click the **Colour icon**, select **Upload an image**, navigate to the folder that contains the application you downloaded from GitHub, select the **color_icon.png** file and click **Open**



46. Click **Save**



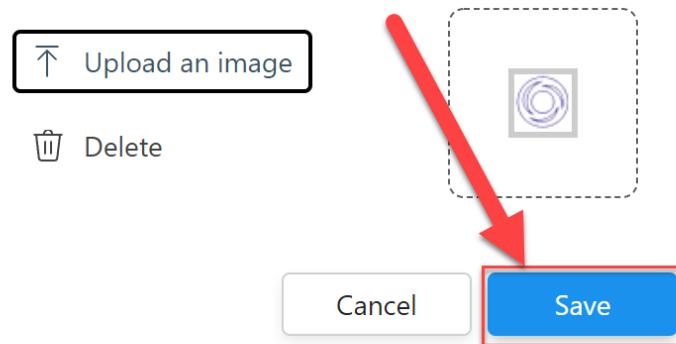
47. Click the **Outline icon**, select **Upload an image**, navigate to the folder that contains the application you downloaded from GitHub, select the **outline_icon.png** file and click **Open**



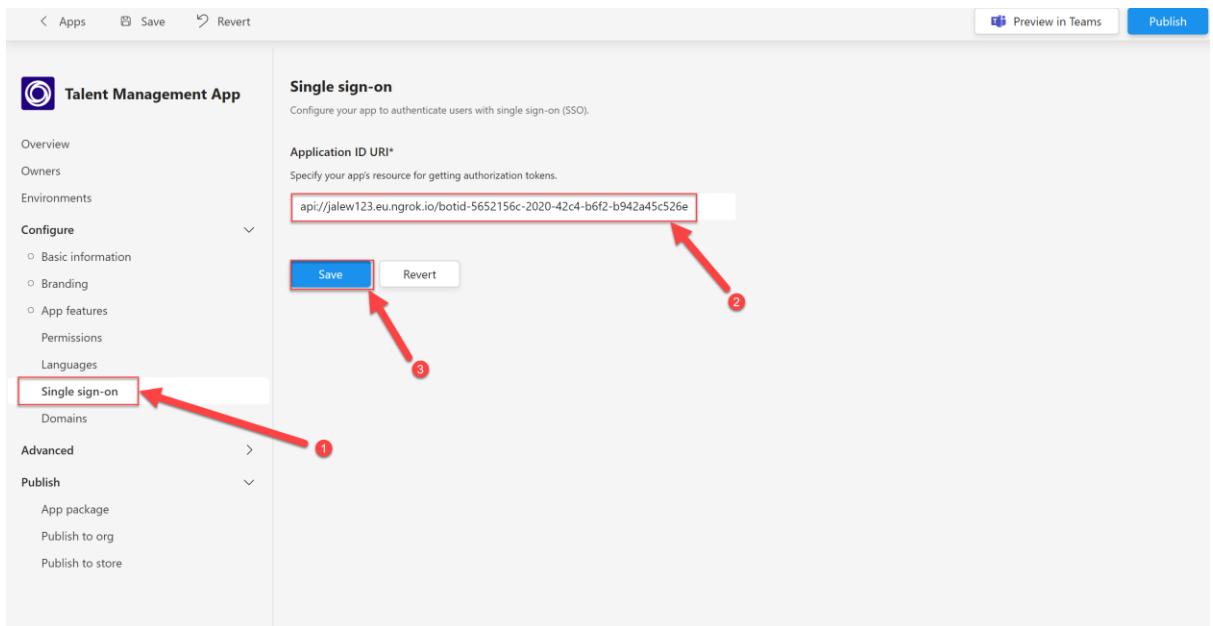
48. Click Save

Outline icon

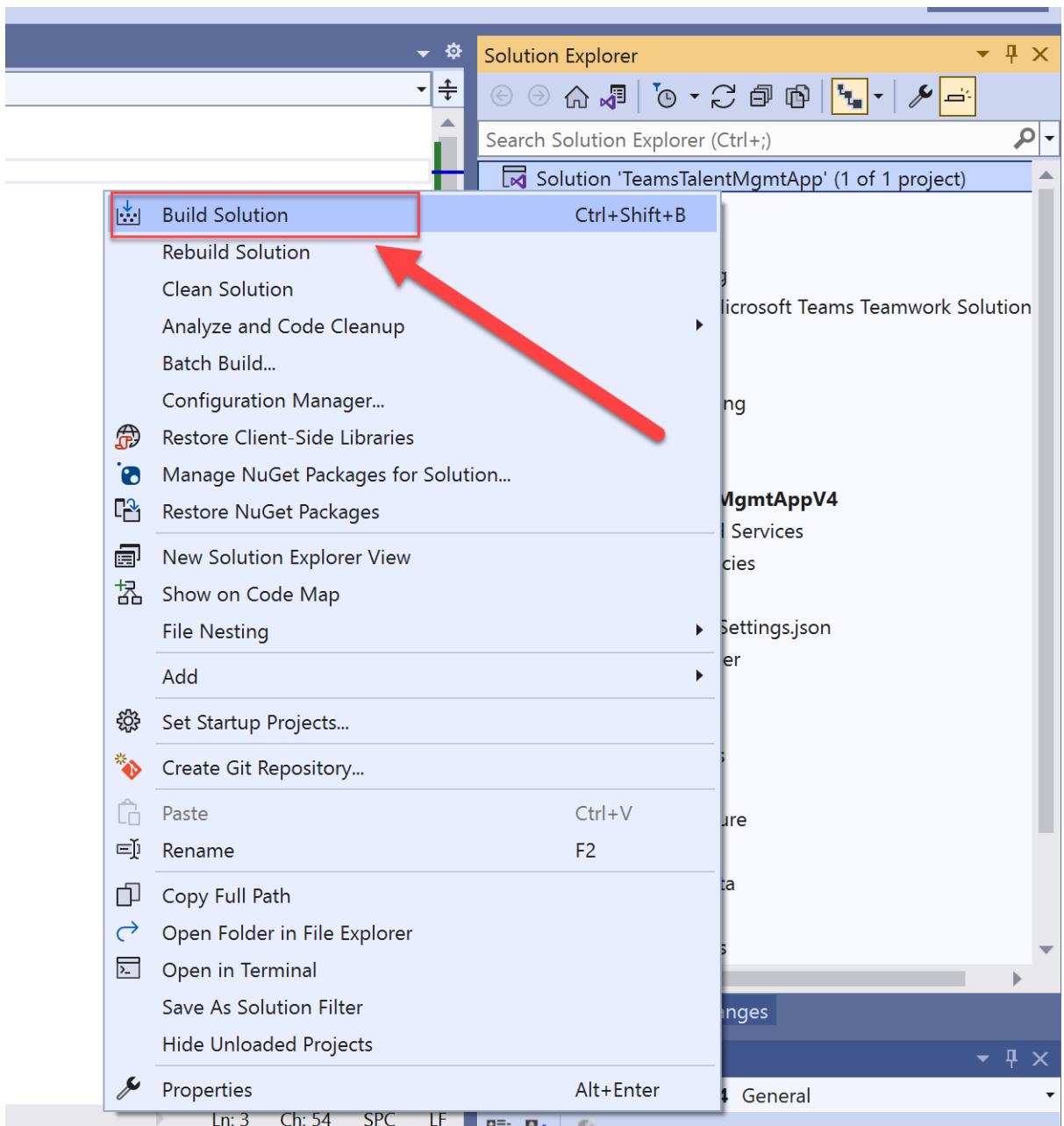
Select a white or transparent 32x32-pixel PNG.



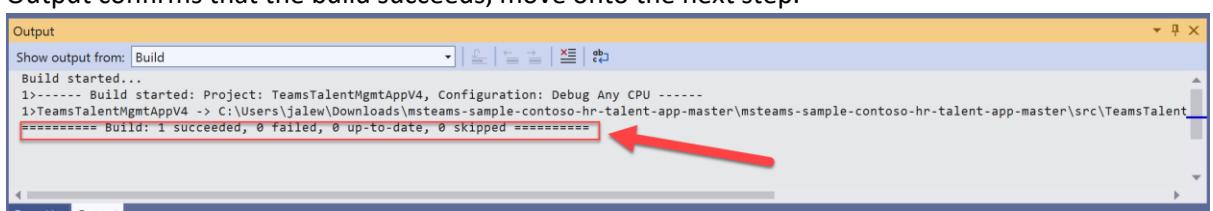
49. The final step to configure in the Manifest, during the initial setup, is the Single sign-on properties. Select **Single sign-on**, enter the **ApplicationIdURI** value into the textbox and click **Save**.



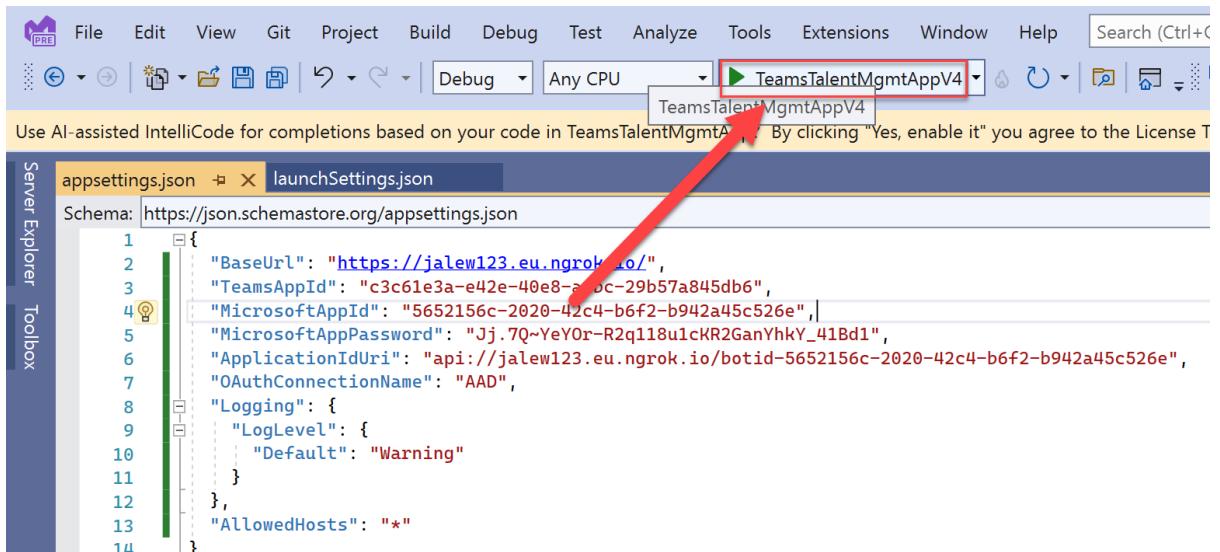
50. Now that the manifest has been configured and all 7 settings in appsettings.json have values associated to them, we are in a position to be able to build and run the app locally, using Visual Studio. **Right click the Solution** and click **Build Solution**.



51. At the bottom of Visual Studio, the Output section will begin populating with text. Once the Output confirms that the build succeeds, move onto the next step.



52. Click the Play button, to run (debug) the Talent Management App locally.



53. All being well, you should receive a message in a console window, confirming that the application is running and listening on ports 5000 and 5001.

```

C:\Users\jalew\Downloads\msteams-sample-contoso-hr-talent-app-master\msteams-sample-contoso-hr-talent-app-master\src\TeamsTalentMgmtAppV4
Hosting environment: Development
Content root path: C:\Users\jalew\Downloads\msteams-sample-contoso-hr-talent-app-master\msteams-sample-contoso-hr-talent-app-master\src\TeamsTalentMgmtAppV4
Now listening on: https://localhost:5001
Now listening on: http://localhost:5000
Application started. Press Ctrl+C to shut down.

```

54. The NGROK tunnel that we established earlier, should be forwarding traffic it receives on 443 (https) to localhost:5000. We can test this by going to the following URL in an InPrivate browser window: **NGROK/StaticViews/OpenPositionsPersonalTab.html** – you should then be sent through an interactive Login and Consent process via Azure AD, and presented with the following dashboard view:

Job Title	Location	Days Open	Applied	Screening	Interviewing	Offered
Senior Software Engineer	New York, NY	75	0	0	3	1
Full Stack Developer	Chicago, IL	40	0	0	1	1
Senior Designer	San Francisco, CA	30	2	0	0	1

Nice job! You have setup the developer environment and Azure resources required to run this application locally. Now that the application is running and working outside of Teams. We can start to configure our Teams App Manifest to bring the apps functionality inside Teams. We'll start with this Open Positions dashboard view in the next lab!

2) Implement a Tab inside a Personal App

In this step we will configure the Positions dashboard to work inside Teams as a Tab inside a Personal App. We will also see Teams SSO in action and review the code that allows this all to work!

1. Go back to the Teams Developer Portal and open your Manifest. Select **App Features**, and then select **Personal app**.

The screenshot shows the Microsoft Teams App Management interface for a 'Talent Management App'. On the left, there's a sidebar with sections like Overview, Owners, Environments, Configure (with sub-options Basic information and Branding), App features (which is highlighted with a red box and arrow 1), Permissions, Languages, Single sign-on, Domains, Advanced, Publish (with sub-options App package, Publish to org, Publish to store), and a preview in Teams button.

The main area is titled 'App features' and contains a heading 'Select a feature to add'. It lists several options:

- Personal app**: A dedicated workspace or bot to help individual users focus on their own tasks or view activities important to them. (This option is highlighted with a red box and arrow 2.)
- Group and channel app**: A space to display hosted app experiences (such as a list or dashboard) in team channels and group chats.
- Bot**: A conversational UI that can perform a set of tasks, reply to questions, and proactively send notifications.
- Connector**: A way to automatically send notifications and messages from your app to a channel.
- Meeting extension**: Options for integrating your app with the Teams meeting experience, including the meeting stage and chat.
- Scene**: A custom virtual scene people can use in their Teams Together mode meetings.
- Activity feed notification**: Keeps users informed and engaged with app notifications in the activity feed.

2. Select **Create your first personal app tab**.

The screenshot shows the 'Personal app' tab creation interface. At the top, it says 'Personal app' and provides a brief description: 'Personal apps are a set of tabs scoped for individual use. These tabs can be like a webpage (e.g., a Home tab) or an area to message a bot (e.g., a Chat tab).'

In the center, there's a decorative icon of a purple folder containing documents, a chart, and a trophy.

At the bottom, there's a large blue button labeled 'Create your first personal app tab'.

3. Enter the name **Positions**, leave the Entity ID as default, Content URL as **NGROK/StaticViews/OpenPositionsPersonalTab.html** and Website URL **NGROK/StaticViews/OpenPositionsPersonalTab.html** – The Website URL is what will be opened if the user wants to open the website outside of Teams. The Content URL is what will be used when the Tab is loaded inside Teams.

Add a tab to your personal app

Define a set of tabs to display in your personal app. An About tab is created automatically by default. [Learn more](#)

Name*

Positions

1

Entity ID*

09b5e445-4113-4924-80af-eef05e80e564

2

Content URL*

<https://jalew123.eu.ngrok.io/StaticViews/OpenPositionsPersonalTab.html>

3

Website URL

<https://jalew123.eu.ngrok.io/StaticViews/OpenPositionsPersonalTab.html>

4

Cancel

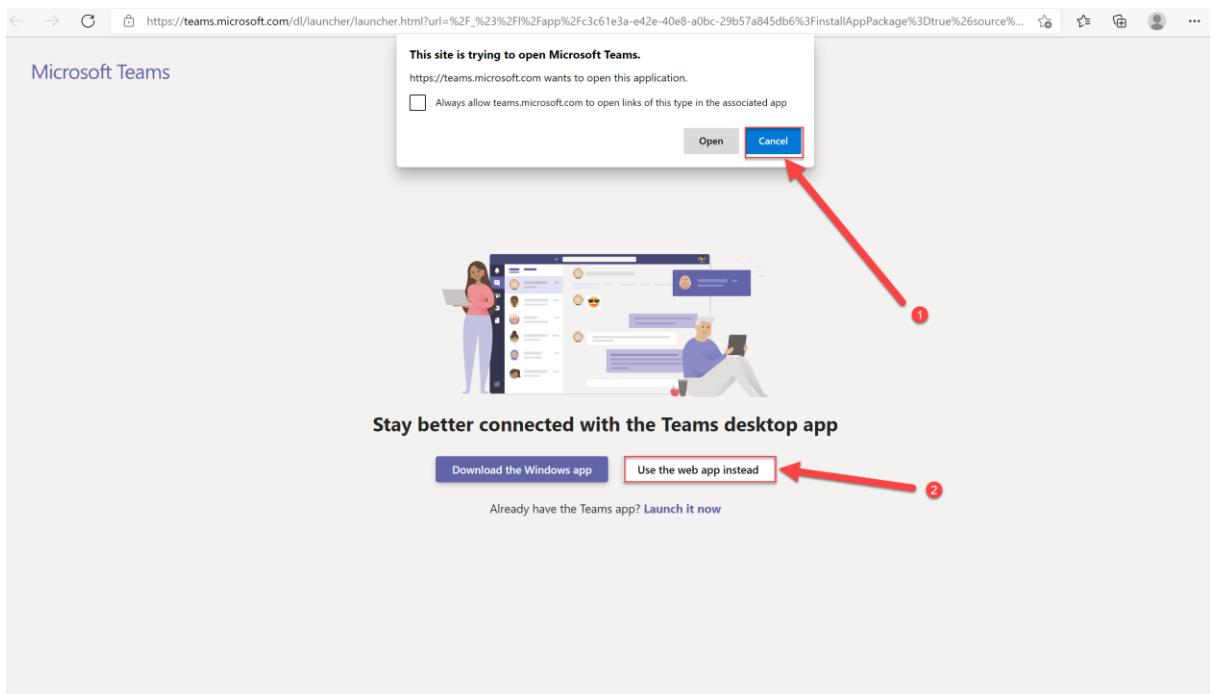
5

Add

4. Click Save and select Preview in Teams

The screenshot shows the Microsoft Teams App Management interface for a 'Talent Management App'. On the left, there's a sidebar with options like Overview, Owners, Environments, Configure (with sub-options like Basic information, Branding, App features, Permissions, Languages, Single sign-on, Domains), Advanced, and Publish. The main area is titled 'Personal app' and contains a table with a single row: Tab name (Positions) and URL (<https://jalew123.eu.ngrok.io/StaticViews/OpenPositionsPersonalTab.html>). Below the table is a button labeled '+ Add a personal tab'. At the bottom of the main area are 'Save' and 'Revert' buttons. In the top right corner, there are 'Preview in Teams' (with a red arrow pointing to it), 'Publish' (disabled), and a three-dot menu. Red numbers 1 and 2 are overlaid on the 'Save' button and the 'Preview in Teams' button respectively.

5. Select Cancel and click Use the web app instead.



6. After a short wait, your app will appear in Teams. Select **Add**.

 Talent Management App

Add

[About](#)

[Permissions](#)

i This app is not from your organisation's app catalogue or the Teams store. Do not proceed to add the app unless you are testing it in development or trust the person who shared it with you.

Talent Management App

Talent Management App

Personal app

Keep track of important content and info

Created by: [Contoso](#)
Version 1.0.0

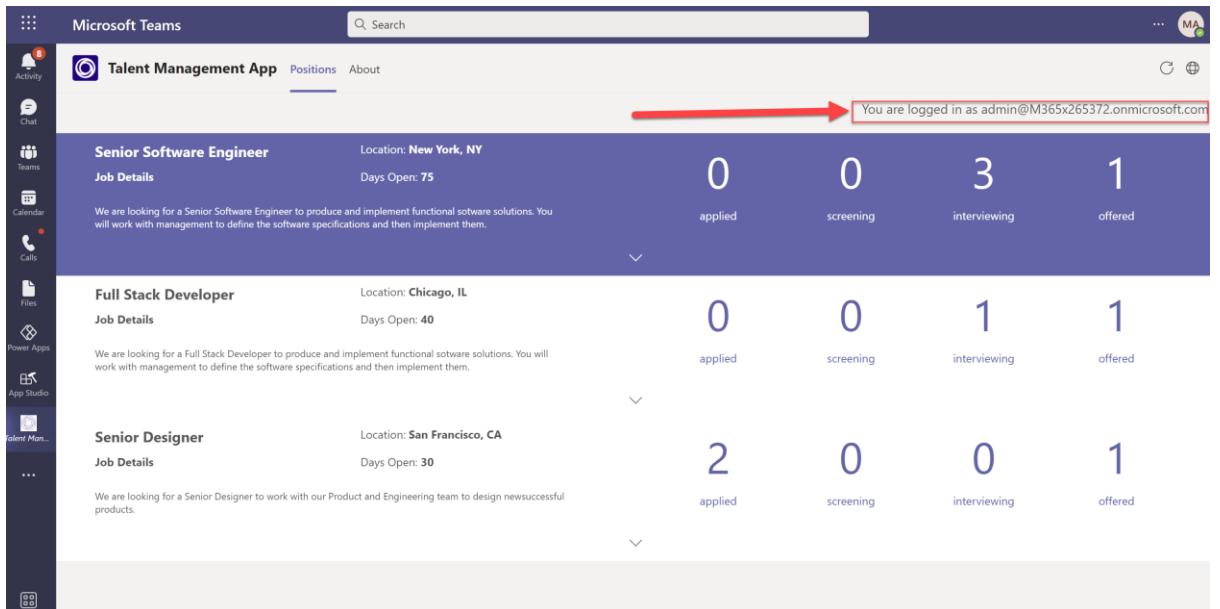
Permissions

This app will have permission to:

- Receive messages and data that I provide to it.
- Access my profile information such as my name, email address, company name, and preferred language.

By using Talent Management App,
you agree to the [privacy policy](#) and
[terms of use](#).

7. Once the Tab has successfully loaded, confirm that you are logged in as the user who is logged into Teams.



Microsoft Teams

Search

Talent Management App Positions About

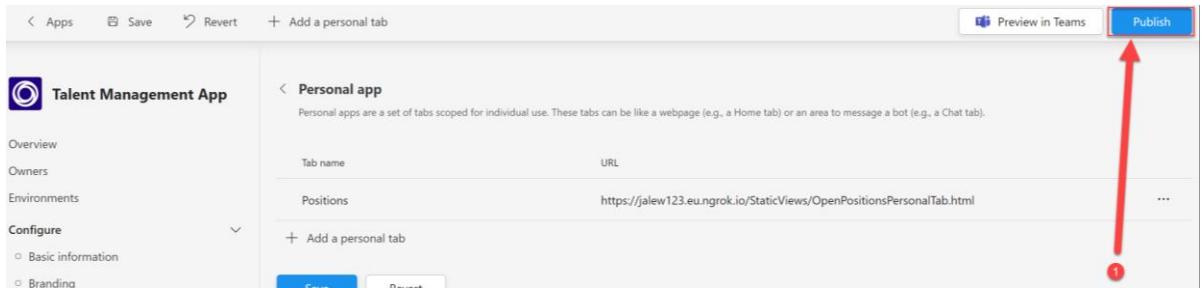
You are logged in as admin@M365x265372.onmicrosoft.com

Senior Software Engineer Location: New York, NY
Job Details Days Open: 75
We are looking for a Senior Software Engineer to produce and implement functional software solutions. You will work with management to define the software specifications and then implement them.

Full Stack Developer Location: Chicago, IL
Job Details Days Open: 40
We are looking for a Full Stack Developer to produce and implement functional software solutions. You will work with management to define the software specifications and then implement them.

Senior Designer Location: San Francisco, CA
Job Details Days Open: 30
We are looking for a Senior Designer to work with our Product and Engineering team to design new successful products.

8. We will now review the App Manifest. Go back to the Teams Developer Portal, open your Manifest and select **Publish**.



Apps Save Revert + Add a personal tab Preview in Teams Publish

Talent Management App

Personal app

Personal apps are a set of tabs scoped for individual use. These tabs can be like a webpage (e.g., a Home tab) or an area to message a bot (e.g., a Chat tab).

Tab name	URL
Positions	https://jalew123.eu.ngrok.io/StaticViews/OpenPositionsPersonalTab.html

+ Add a personal tab

Save Revert

9. Click **Download the app package**.

Publish your app



Download the app package



Download a copy of your app package, which is specific to your selected environment. Use the package to upload your app in Teams or publish later.



Publish to your org

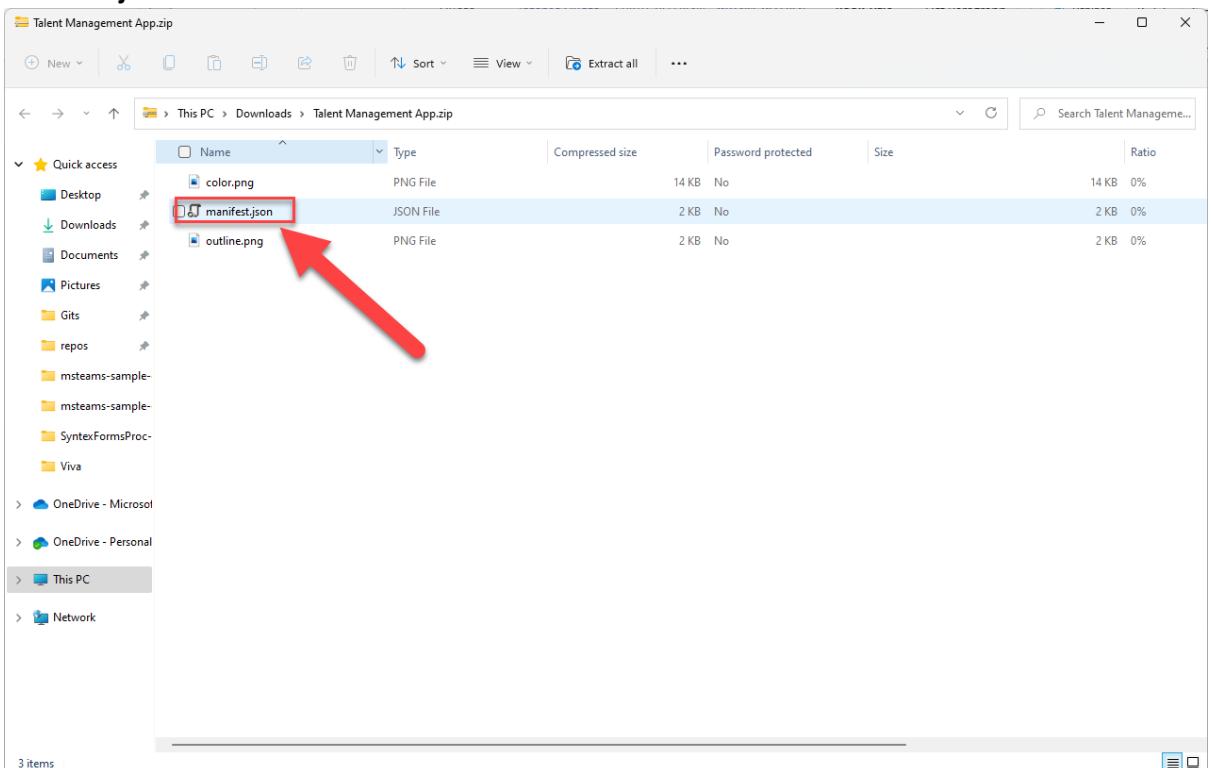
Submit a request to your IT admin to publish your app. It will appear in the Built for your org section of the store once it's approved.



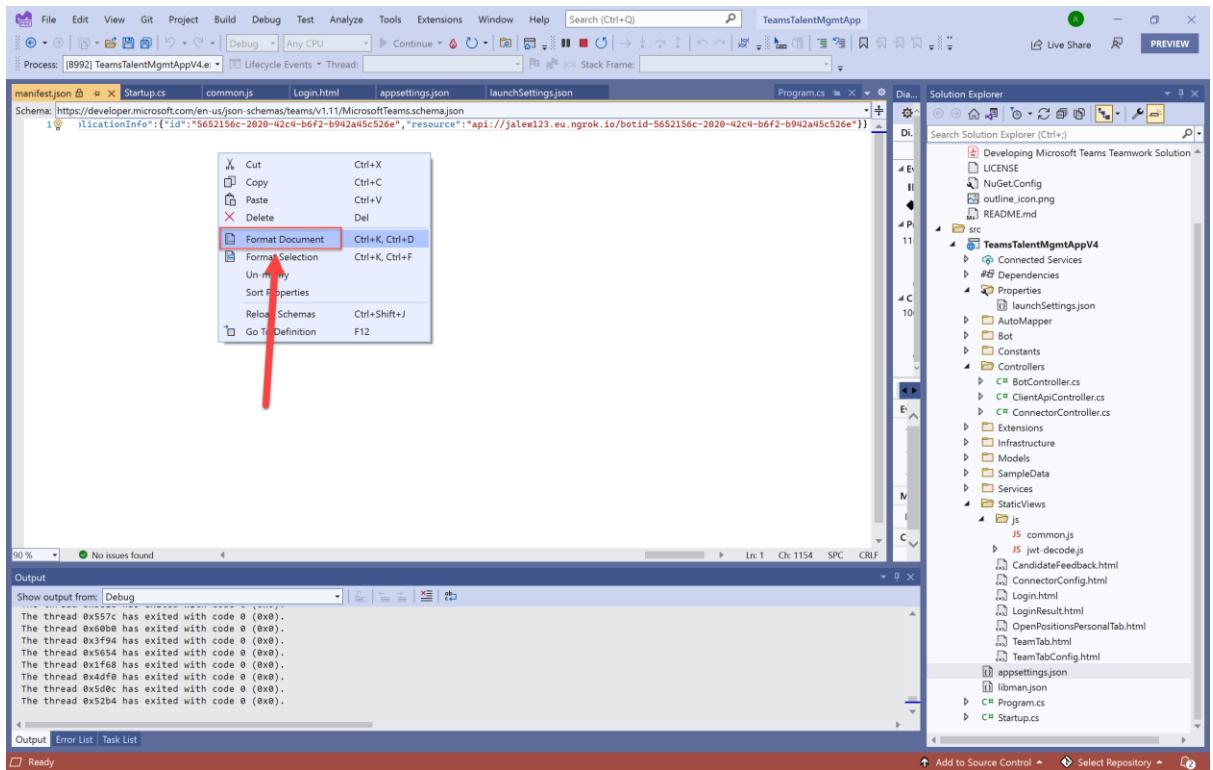
Publish to the Teams store

Make your app available to Teams users everywhere. This option requires Microsoft approval.

10. Open the ZIP that contains your manifest that has just been downloaded and open the **manifest.json** file in Visual Studio.



11. To improve readability, right click the whitespace in the manifest.json file and select **Format Document**.



12. If prompted, select **Ok**.



13. Review the settings and values in this manifest – the manifest is your Teams app! Firstly, see the name, developer and description settings and values. They contain the Basic Information you entered on the first page when you created your Teams app in the Teams Developer Portal. Secondly, review the Static Tabs array – this contains the contentUrl and websiteUrl settings and values. Finally, the webApplicationInfo settings and values contain your Azure AD and SSO information, that will be used when the page is loaded to perform Single Sign-On. This is explained in more detail in the next few steps.

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50

```

No issues found

Ln: 51 Ch: 2 SPC CRLF

14. Go back into Visual Studio, expand the Static Views folder and select the **OpenPositionsPersonalTab.html** file. Review the scripts that are used when this page is loaded, importantly the Teams SDK (line 7) and the common.js (line 11) are used here. The Teams SDK is used in web applications to enable interoperability between the Teams client and the web-app. It can be used to pass context, perform SSO and many other things. common.js is where all the sign-in and Teams related JavaScript resides. We will review this later.

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27

```

Search Solution Explorer (Ctrl+)

- Properties
- AutoMapper
- Bot
- Constants
- Controllers
- Extensions
- Infrastructure
- Models
- SampleData
- Services
- StaticViews
 - JS
 - common.js
 - jwt-decode.js
 - CandidateFeedback.html
 - ConnectorConfig.html
 - Login.html
 - LoginResult.html
 - OpenPositionsPersonalTab.html

15. For this specific page, we are using jQuery to control when the JavaScript functions are called. This is detailed on line 262, where we call and await the completion of **handlePageLoad** (which is found in common.js and handles authentication both inside and outside of Teams), before we call **loadPage** (which is what builds the page and calls APIs to get data for Positions).

```

235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271

```

```

    if (activeIndex !== 0) { newIndex -= 1; }
    break;

    case 40:
      if (activeIndex === maxActiveIndex) { newIndex += 1; }
      break;

    default: return;
  }
  clickOnMainBlock(`#${'main-block-' + newIndex}`);
}

function clickOnMainBlock(currentObject) {
  var index = currentObject.data("index");
  if (index === activeIndex) {
    $(`.active`).removeClass("active");
    currentObject.addClass("active");
  }
  activeIndex = index;
}

function getQueryStringValue(key) {
  return decodeURIComponent(window.location.search.replace(new RegExp(`^.*${key}=[\&\?]*` + encodeURIComponent(key)).replace(/[\.\+\*\?]/g)));
}

$(async () => {
  await this.handlePageLoad();
  this.loadPage();
})

```

</script>

</head>

<body dir="ltr">

16. Expand the **js** folder, found inside **StaticViews**, and open the **common.js** file. Here you can see the **handlePageLoad** function. We are using this to call other functions that then determine if the page is being loaded inside or outside of Teams. If the page is loaded inside Teams, then the function **signInWithTeams** is called, which then attempts to do a Teams SSO. If the page is outside of Teams, the function **outsideTeamsSignIn** is called, which will redirect the user to Azure AD. It's important to note that Azure AD (and many other identity providers) do **not** support iFraming of their authentication pages. As website inside Teams are iFramed, this means that we need a custom way of handling SSO. Hence why the flow of SSO is different when the page is loaded in Teams.

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17

```

```

  handlePageLoad = async () => {
    try {
      window.localStorage.removeItem("isTeams");
      await this.initialiseTeams();
      this.inTeams = true;
    } catch {
      this.inTeams = false;
    }

    if (this.inTeams) {
      await this.signInWithTeams();
    } else {
      outsideTeamsSignIn();
    }
}

```

17. Scroll down and review the **getTeamsToken** function. This is the function that attempts to do a Teams SSO, by calling the **microsoftTeams.authentication.getAuthToken** function. This will take the Access Token that was used to sign-in to Teams, by the end user, and submit it to a back-end service that is operated by Microsoft. The back-end service will then try to swap the Teams Access Token, for an access token that has an Audience of your application. This should work based on the configured that you applied during the setup of the Azure AD App Registration, and will return an access token, with your MicrosoftAppId as the Audience in the claims of the access token, with a scope of `access_as_user`. **If successful, it will set the access token as an item in Local Storage for the Browser**, this can then be used by the page to call protected APIs to get Position and Candidate data. We will inspect this later on in the lab. The **teamsFallbackAuth** function is called if the SSO function fails, and will allow the user to perform an interactive sign-in to either provide Consent, or get an access token for use within the App.

```

getTeamsToken = () => {
    window.localStorage.setItem("isTeams", "yes");
    return new Promise((resolve, reject) => {
        microsoftTeams.authentication.getAuthToken({
            successCallback: (token) => {
                window.localStorage.setItem("userToken", token);
                resolve(token);
            },
            failureCallback: (reason) => {
                reject(reason);
            }
        });
    });
}

function teamsFallbackAuth() {
    microsoftTeams.authentication.authenticate({
        url: window.location.origin + "/StaticViews/Login.html",
        width: 600,
        height: 535,
        successCallback: function (result) {
        },
        failureCallback: function (reason) {
            handleAuthError(reason);
        }
    });
}

```

18. Scrolling to the bottom of the common.js file, you will see the **outsideTeamsSignIn** function. This checks for the existence of an access token in Local Storage, and if one doesn't exist, will call the **authRedirect** function. **authRedirect** will set some properties and then redirect the end-user to **Login.html**, which will then redirect the user to Azure AD for sign-in. Eventually, this will provide the user with an access token and they will return to the OpenPositionsPersonalTab.html page, but this time, the accessToken variable will contain an access token and the user will not be redirected to the Azure AD. **getInfoFromToken** is called on the OpenPositionsPersonalTab.html page, and is used to pull out the claims (specifically the username attribute) for use when that page is loaded.

The screenshot shows the Visual Studio IDE with the file `common.js` open. The code handles user sign-in and redirection. Red boxes highlight three sections of code:

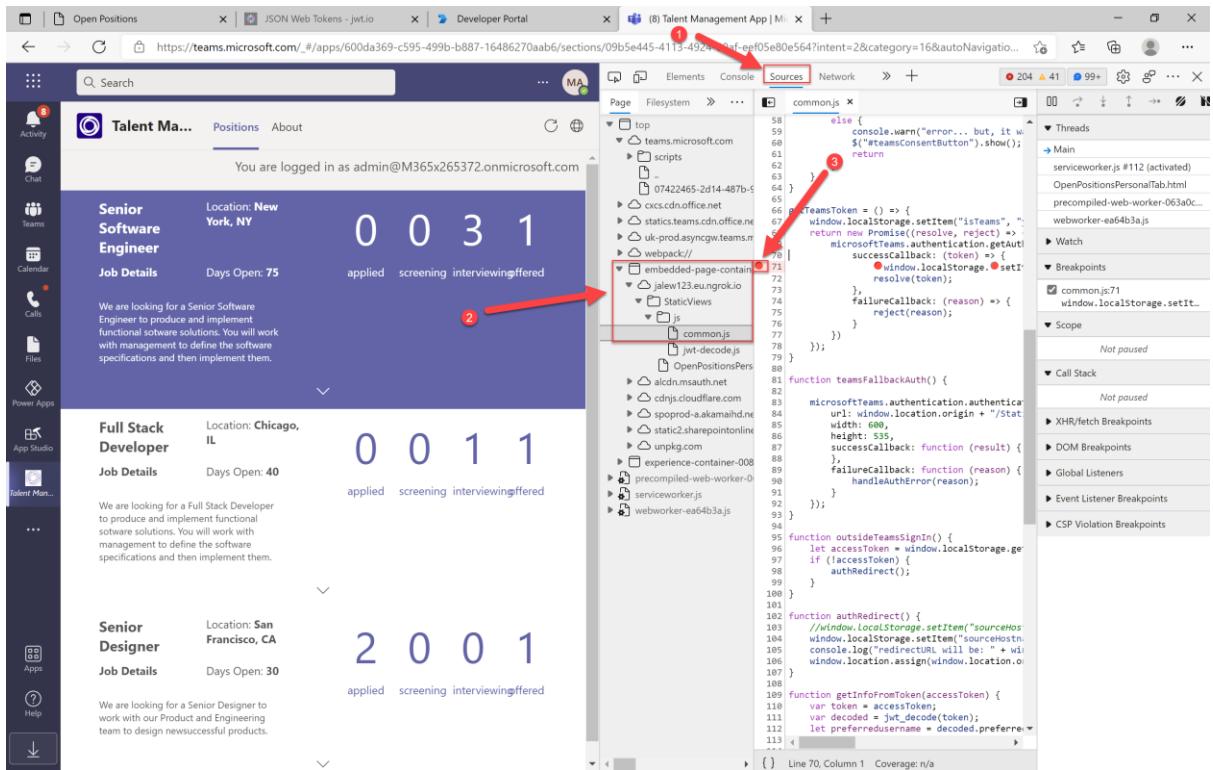
- Line 95: `window.localStorage.setItem("userToken", accessToken);`
- Line 103: `window.localStorage.setItem("sourceHostname", location.protocol + "//" + location.hostname + ":" + location.port + "/StaticViews/OpenPositionsPersonalTab.html");`
- Line 111: `window.location.assign(window.location.origin + "/StaticViews/Login.html");`

19. Go back to **OpenPositionsPersonalTab.html** and review the **loadPage** function. This will call the **getData** function. **getData** uses fetch within JavaScript to make a HTTP request, with the access token obtained from Azure AD in the header, to the apps Client APIs, this will return information about the available positions stored within the app. Once we have this information, and a 200 OK response is returned form the Client APIs, **buildPage** will be called, and the position data (stored in the **json** variable), recruiterId (which is your username) and positionId (which is obtained from the URL querystrings) is passed into this function – **buildPage** will then render the page as you see it!

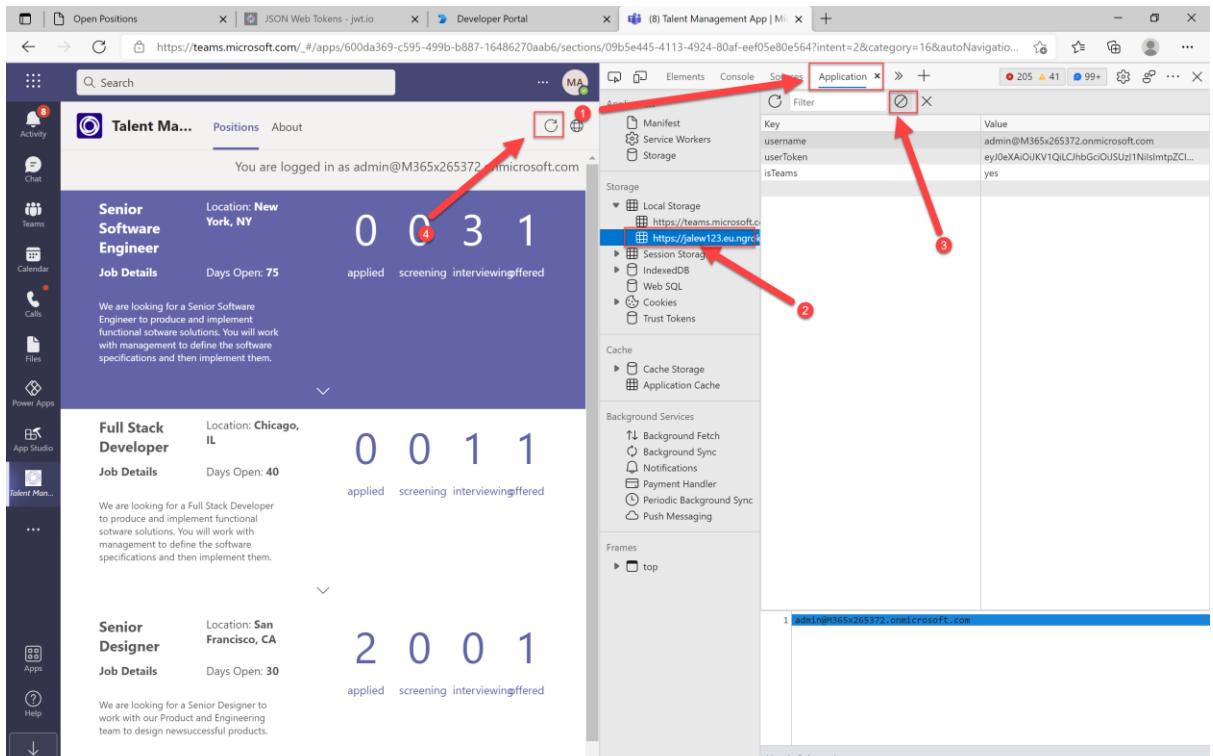
The screenshot shows the Visual Studio IDE with the file `OpenPositionsPersonalTab.html` open. The code contains the **loadPage** function which makes an asynchronous call to the **getData** function. Red boxes highlight three parts of the code:

- Line 107: `async function loadPage() {`
- Line 124: `let data = await fetch(window.location.origin + apiPath, {`
- Line 131: `let json = await data.json();`

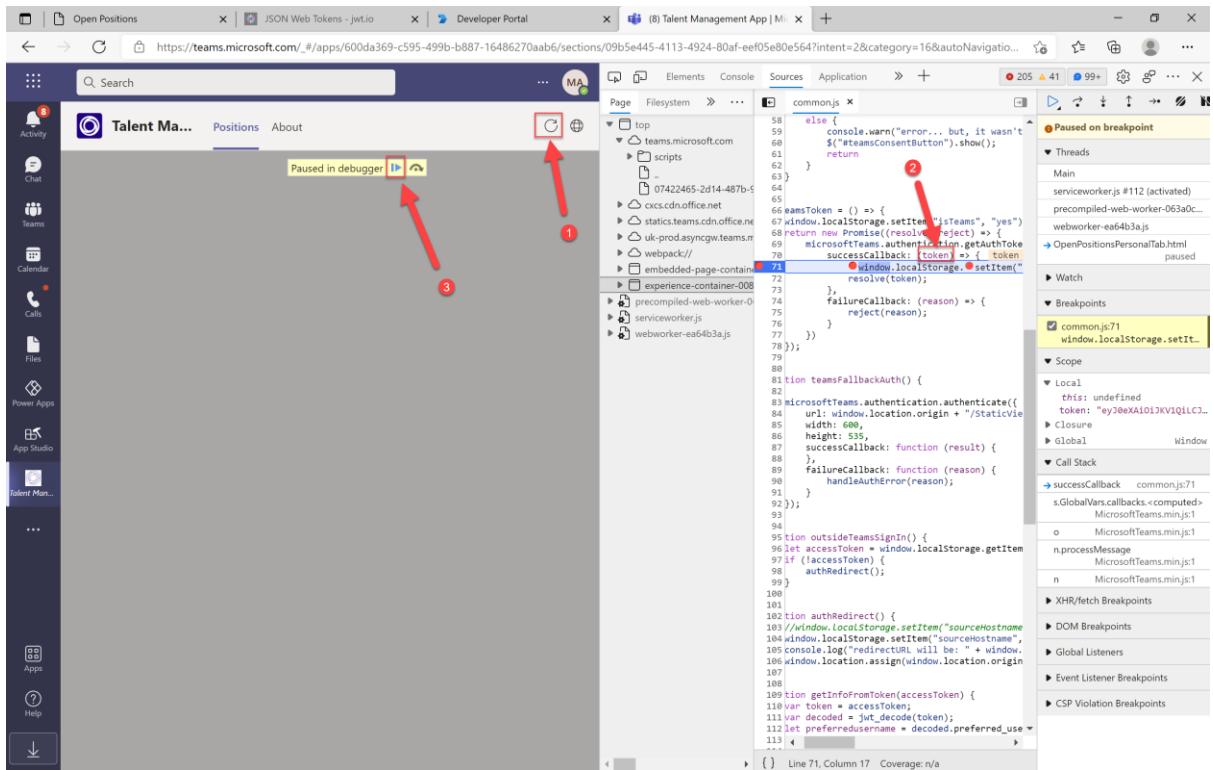
20. Let's see this in action! Go back into Teams and open the Talent Management App. Hit **F12** on your keyboard to open your browser's development tools. Select **Sources**, expand the folder hierarchy, until you find **common.js** (which is inside **StaticViews/js** folder, found within embedded-page/NGROK). Scroll down and click **to the left of line 71 to add a Breakpoint** (this will add a red dot to the left of the number).



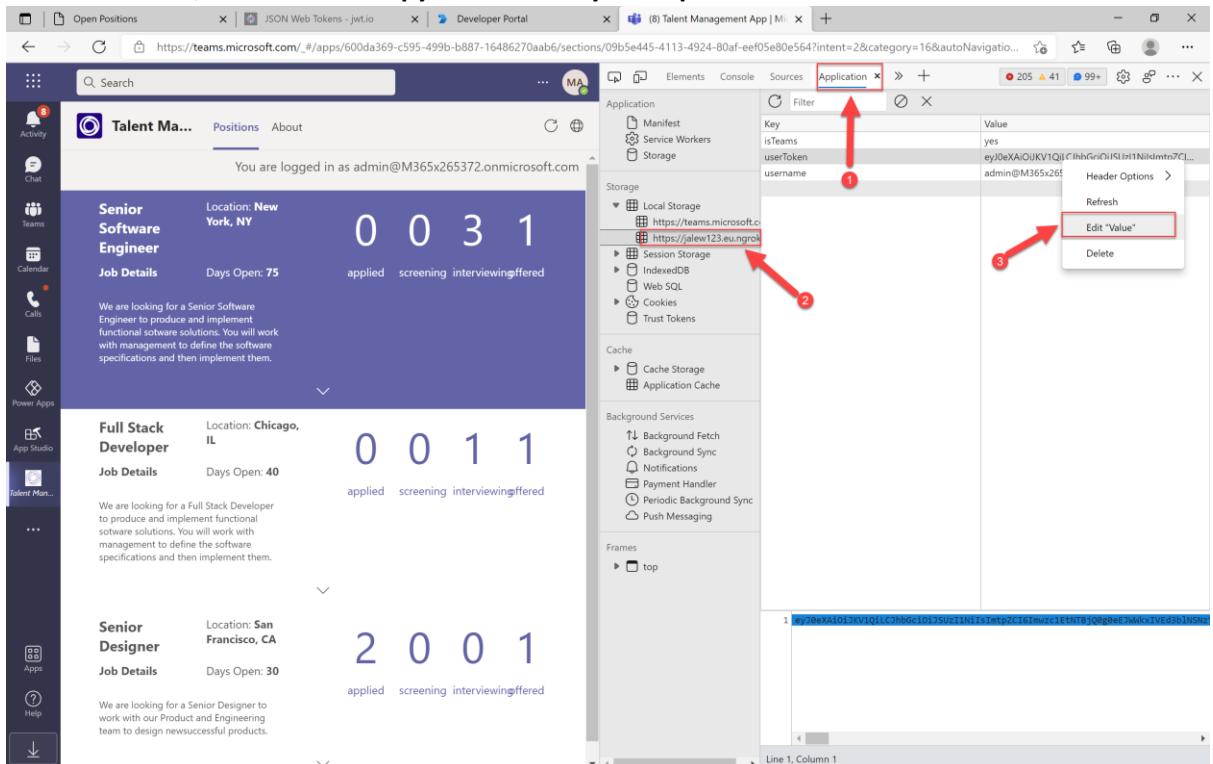
21. Before we re-load the page, it's a good idea to remove the items in Local Storage for this app. Click **Application**, expand **Local Storage** and select your NGROK app. Click the **clear** button.



22. Hit the **Reload page** button. This will trigger the breakpoint. Mouse over the **Token** variable on line 70. This should then present you with an Access Token that has been retrieved via the Teams SSO method. Hit the **Continue** button to allow the page to complete loading.



23. Go back to the **Application** tab and select your app under **Local Storage**. Right click the **userToken** item, click **Edit** and copy the value to your clipboard.



24. Open a browser tab and go to <https://jwt.io>, paste your access token into the left input box and review the claims within the token. Importantly, you will see that it was issued by Azure AD, the Audience is your MicrosoftAppId, the scope is access_as_user & the user is the current logged in Teams user.

The screenshot shows a JWT token being decoded on jwt.io. The token is pasted into the 'Encoded' field. The 'Decoded' field shows the token structure with two red arrows pointing to specific parts:

- Header:** ALGORITHM & TOKEN TYPE
- PayLoad:** DATA

```

1
{
  "typ": "JWT",
  "alg": "RS256",
  "kid": "13s0-58cCH4xBVZLHTGwNsr7680"
}

2
{
  "aud": "5652156c-2028-42c4-b6f2-b942a45c526e",
  "iss": "https://login.microsoftonline.com/d06ea5c6-1047-45d0-8ea9-7de7d40e3c58/v2.0",
  "iat": 1633704887,
  "nbf": 1633704887,
  "exp": 1633712978,
  "aio": "ATQAY/8TAAAAR8oXctsD6D1B7DBbf10WIaaN6vx5bqqXtdF79hYQy4pe9g96a1319aNg/SVW10b",
  "azp": "5e3ce6c0-2b1f-4285-8d4b-75ee78787346",
  "azpacr": "0",
  "name": "MOD Administrator",
  "oid": "0af4b8ce-b300-4ca9-992b-cd26956d7ad9",
  "preferred_username": "admin@M365x265372.onmicrosoft.com",
  "rh": "0.ARoAxqVu0EcQ0EW0qX3n1A48NM0mpF4fK4VCjUt17nh4c0Z5AIw",
  "scp": "access_as_user",
  "sub": "819n4kD0yfQ0YNvpaXuea2czvnrivxZ17WRj1P1t-Ag",
  "tid": "d06ea5c6-1047-45d0-8ea9-7de7d40e3c58",
  "uti": "LijSBxDN00STS167aiVKAA",
  "ver": "1"
}

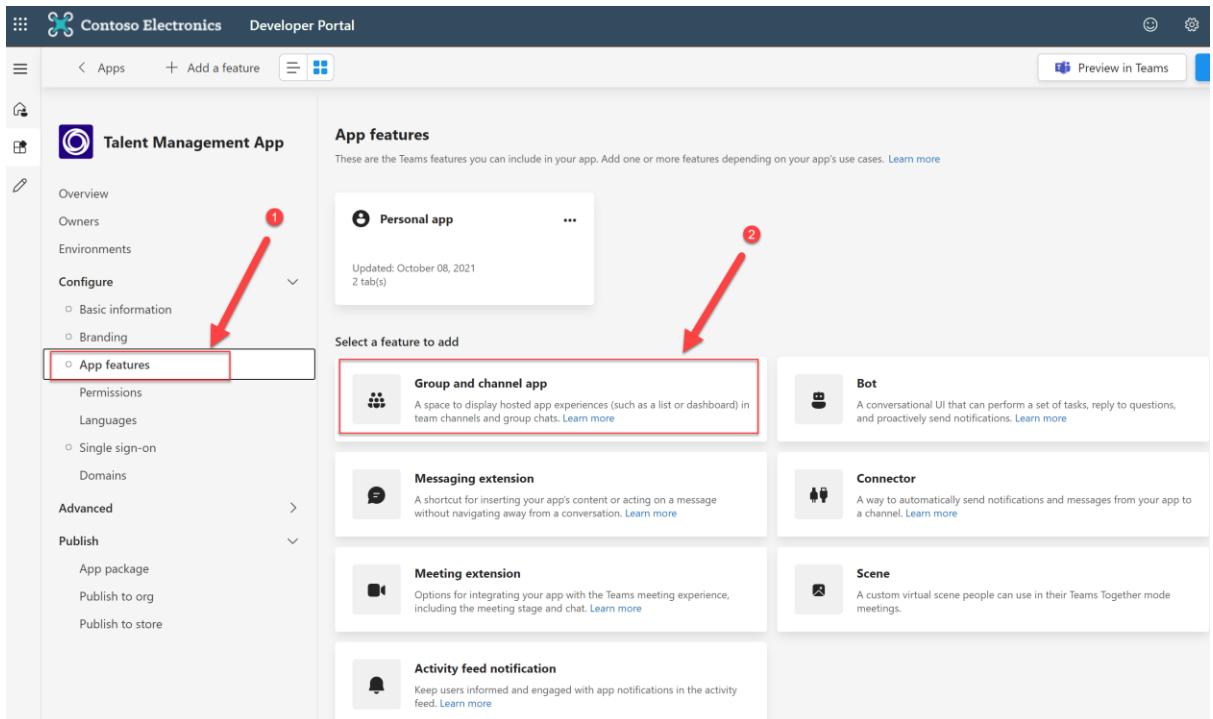
```

Great work! Now that the Personal Tab is working, and we have seen Teams SSO in action, let's move onto a Tab in a Team! These work slightly differently as they use a configuration page.

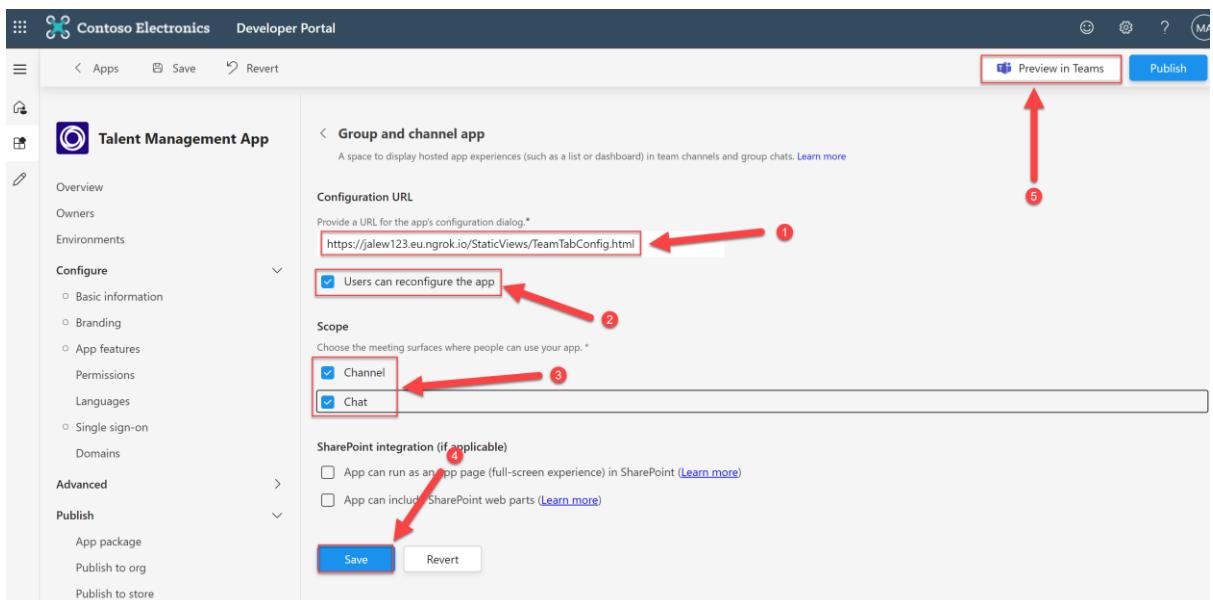
3) Implement a Tab inside a Team Channel

In this step we will implement a Team Channel Tab, which uses a special Configuration page, that we will inspect during this section of the lab. Channel Tabs are great for pinning specific information about specific objects that exist within your application and enable collaboration around that specific object. In this example, we'll be pinning a tab that contains details about an available position and the associated candidates that have applied for that position.

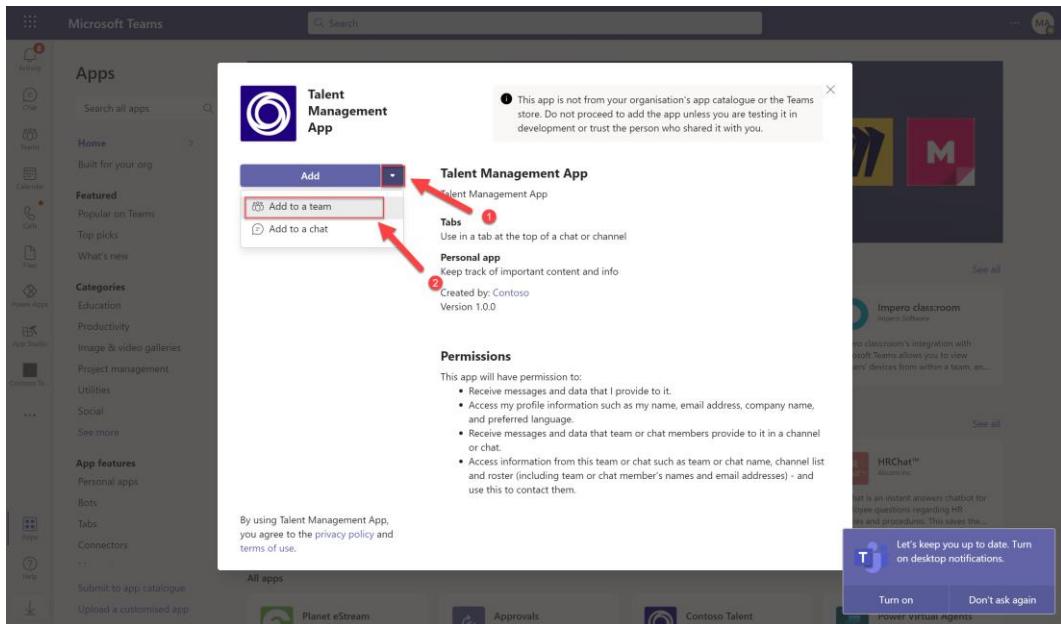
1. Go back to the Teams Developer Portal and open the Teams Talent Management App. Select App Features and click Group and channel app.



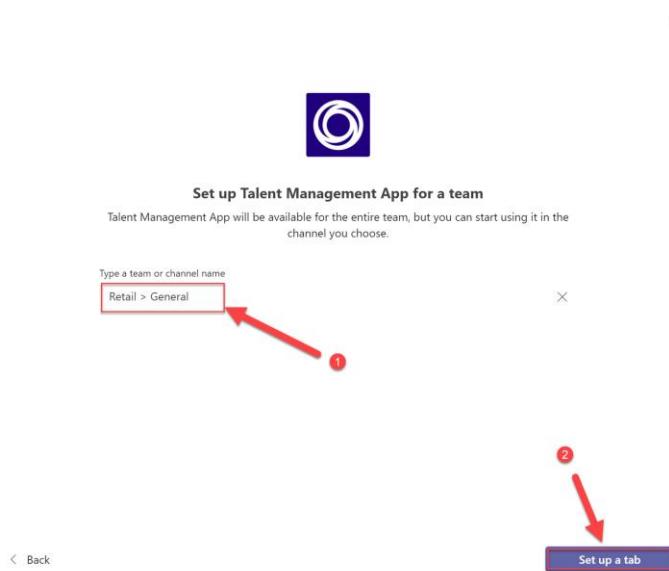
- Enter the following into the Configuration URL **NGROK/StaticViews/TeamTabConfig.html** – select **Users can reconfigure the app** and select both **Channel** and **Chat** in the scopes. Click **Save**. For context, the Configuration URL is the URL that will open the Tab Configuration page, this is a special page that allows users to decide which Position you want to include in the Team Tab, we will inspect the JavaScript later in this lab that enables this. Click **Preview in Teams**.



- Select the **drop down button** and select **Add to a Team**.



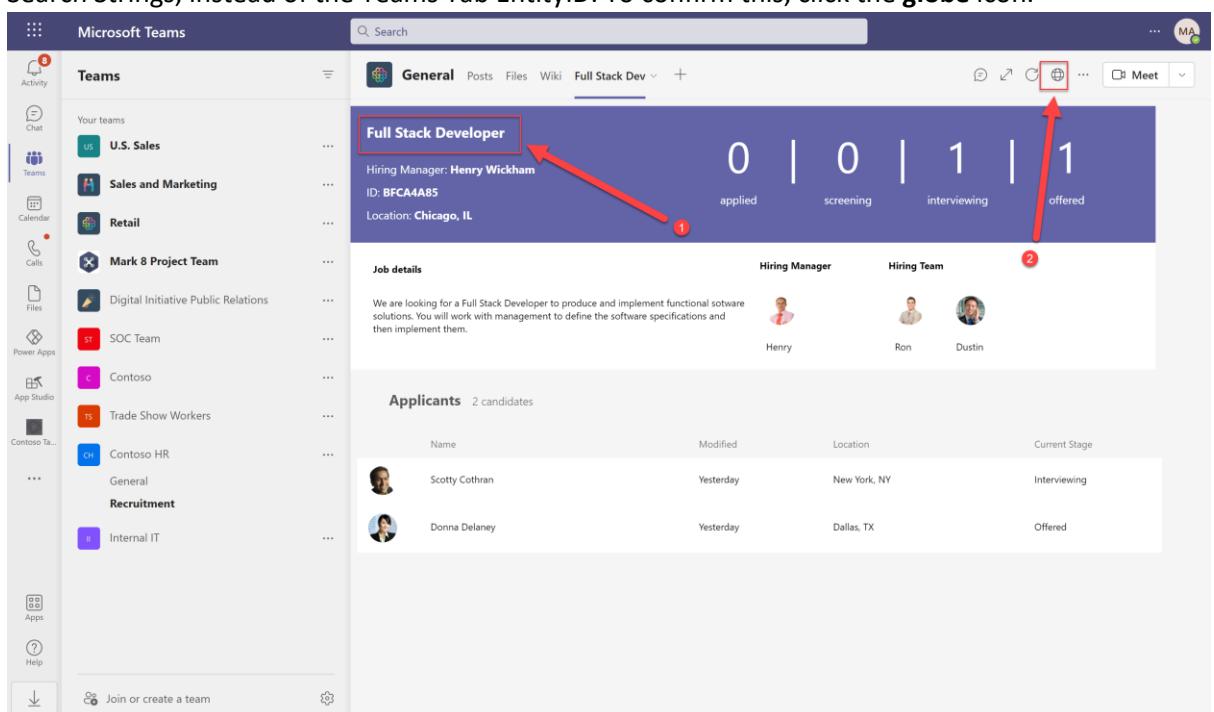
4. Select a Team and Channel, in this demo I selected **Retail -> General**, and click **Set up a tab**.



5. This will open the Tab Configuration page defined in the App Manifest. On this page you can enter a name for the Tab and select a job Position. Enter the tab name of **Full Stack Dev** and select **Full Stack Developer** in the dropdown. Click **Save**.



6. This should load the TeamTab.html page inside the Team Channel Tab, that contains information about the Full Stack Developer position. This is achieved by passing the Teams Tab EntityID to our web-application, via the microsoftTeams.getContext function in the Teams SDK. This page has also been configured to load outside of Teams, by using the URL Search Strings, instead of the Teams Tab EntityID. To confirm this, click the **globe** icon.



7. You will then see the page load successfully, and if you review the URL Search Strings, you will see **positionID & web** contain values.

Full Stack Developer

Hiring Manager: Henry Wickham
ID: BFCA4A85
Location: Chicago, IL

Job details

We are looking for a Full Stack Developer to produce and implement functional software solutions. You will work with management to define the software specifications and then implement them.

Hiring Manager

Henry

Hiring Team

Ron Dustin

Applicants 2 candidates

Name	Modified	Location	Current Stage
Scotty Cothran	Yesterday	New York, NY	Interviewing
Donna Delaney	Yesterday	Dallas, TX	Offered

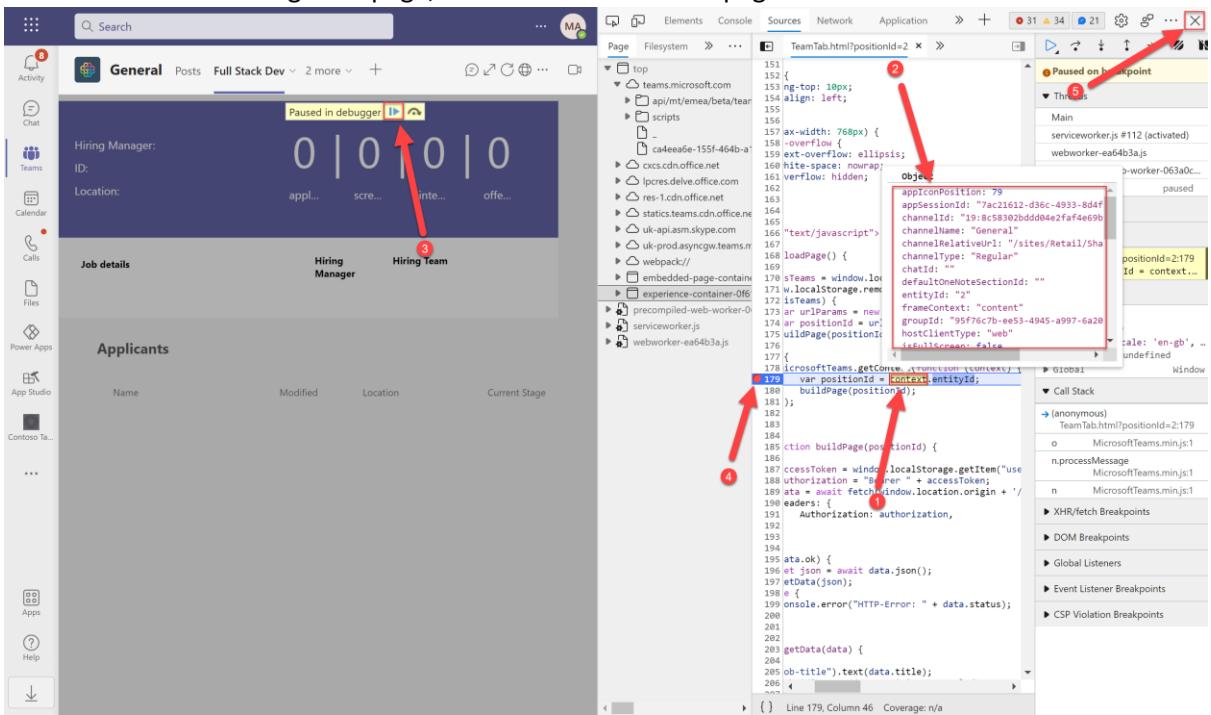
- Let's review the code that enables this functionality. Go back into Teams, within the Channel Tab. Press F12 to open the development tools. Select Sources, and expand embedded-page, NGROK, StaticViews and select TeamTab.html. Scroll down to line 177 in the code and review the microsoftTeams.getContext function, this is the capability within the Teams SDK that will pass the EntityID, and a lot of other context information through to the Talent Management Web App. We use the EntityID, stored within the Context variable, to build the page by calling buildPage(positionID) on line 180. Add a breakpoint to line 179, as we will use this to review the information stored within Context. Refresh the page.

```

1. media (max-width: 768px) {
2.   .text-overflow {
3.     text-overflow: ellipsis;
4.     white-space: nowrap;
5.     overflow: hidden;
6.   }
7. }

8. <script type="text/javascript">
9. 
10. function loadPage() {
11.   let isTeams = window.localStorage.getItem("isTeams");
12.   if (!isTeams) {
13.     var urlParams = new URLSearchParams(window.location.search);
14.     var positionId = urlParams.get("positionId");
15.     buildPage(positionId);
16.   }
17. }
18. 
19. microsoftTeams.getContext(function(context) {
20.   let accessToken = window.localStorage.getItem("accessToken");
21.   if (accessToken) {
22.     let authorization = "Bearer " + accessToken;
23.     headers: {
24.       Authorization: authorization
25.     }
26.   }
27.   if (data.ok) {
28.     let json = await data.json();
29.     getData(json);
30.   } else {
31.     console.error("HTTP-Error: " + data.error);
32.   }
33. }
34. 
35. function getData(data) {
36.   $("#job-title").text(data.title);
37.   $("#job-id").text(data.positionId);
38. }
39. 
```

9. The breakpoint will be triggered. **Mouse over the word context** on line 179, this will display the values stored within the context variable, which are set when **microsoftTeams.getContext** is called. You can see here that a lot of useful information is available to you, when you are building your web-app. In our case, we simply use **entityId** to decide which position we should show on the page, but this context can be used to discover **channelName**, **themeSettings**, and a load of other settings! Review this and consider what you could use in your application to provide a brilliant experience when your app is delivered inside Teams. Click the **continue** button to resume the debugger, remove the **breakpoint** by clicking red circle icon to the left of line 179, and **close the developer tools**. Let's now go back into visual studio and review the steps that are taken to pass the **entityId** from the **TeamTabConfig.html** page, into the **TeamTab.html** page.



10. Open Visual Studio, and open the **TeamTabConfig.html** page, scroll down to line 184 and review the code contained in the **rebuildDropdown** and **updateSelectedPosition** functions. These work together to build the dropdown with available Positions, and when a position is selected, they set the **currentSelectedPositionId** variable with the ID of the selected position. Scrolling further down to line 227, review the **microsoftTeams.settings.registerOnSaveHandler** function – this uses the **currentSelectedPositionId** variable, to set the **entityId** that will be passed into the **TeamTab.html** page when it is loaded inside Teams.

The screenshot shows the Visual Studio IDE with two code editors open. The top editor contains `TeamTabConfig.html` and the bottom editor contains `Startup.cs`. Red arrows point from specific lines of code in both files to numbered callouts in the Solution Explorer pane.

TeamTabConfig.html (Top Editor)

```

    ...
    191     currentSelectedPositionId = positionId;
    192     $("#currentPosition").text(positionName);
    193     $("#positionsDropdownContent").width($("#posting").outerWidth());
    194     checkValidityState();
    195   });
    196 }
    197
    198 function updateSelectedPosition(positionId, positionName) {
    199   currentSelectedPositionId = positionId;
    200   $("#currentPosition").text(positionName);
    201   $("#positionsDropdownContent").width($("#posting").outerWidth());
    202   checkValidityState();
    203 }
    204
    205 //fetch stuff
    206
    207 let accessToken = window.localStorage.getItem("userToken");
    208 let authorization = "Bearer " + accessToken;
    209
    210 //data = await fetch(window.location.origin + "/api/positions");
  
```

Startup.cs (Bottom Editor)

```

    ...
    218 if (data.ok) {
    219   let json = await data.json();
    220   updateSelectedPosition(json[0].positionId, "ID " + json[0].positionExternalId + " - " + json[0].title);
    221   positionData = json;
    222   rebuildDropdown();
    223 } else {
    224   console.error("HTTP-Error: " + data.status);
    225 }
    226
    227 //Save handler when user clicked on Save button
    228 microsoftTeams.settings.registerOnSaveHandler(function (saveEvent) {
    229
    230   microsoftTeams.getContext(function () {
    231     var name = nameSelector.val();
    232     var url = window.location.origin + '/StaticViews/TeamTab.html?positionId=' + currentSelectedPositionId;
    233     var websiteUrl = url + '&web=1';
    234
    235     console.log(nameSelector.val());
    236     console.log(url);
    237     console.log(websiteUrl);
    238
    239     microsoftTeams.settings.setSettings({
    240       entityId: currentSelectedPositionId,
    241       contentUrl: url,
    242       suggestedDisplayName: name,
    243       websiteUrl: websiteUrl
    244     });
    245
    246     saveEvent.notifySuccess();
    247   });
    248 }
    249
    250
    251
    252
    253
  
```

Solution Explorer (Right)

The Solution Explorer shows the project structure for `TeamsTalentMgmtAppV4`. Red callouts point to specific files:

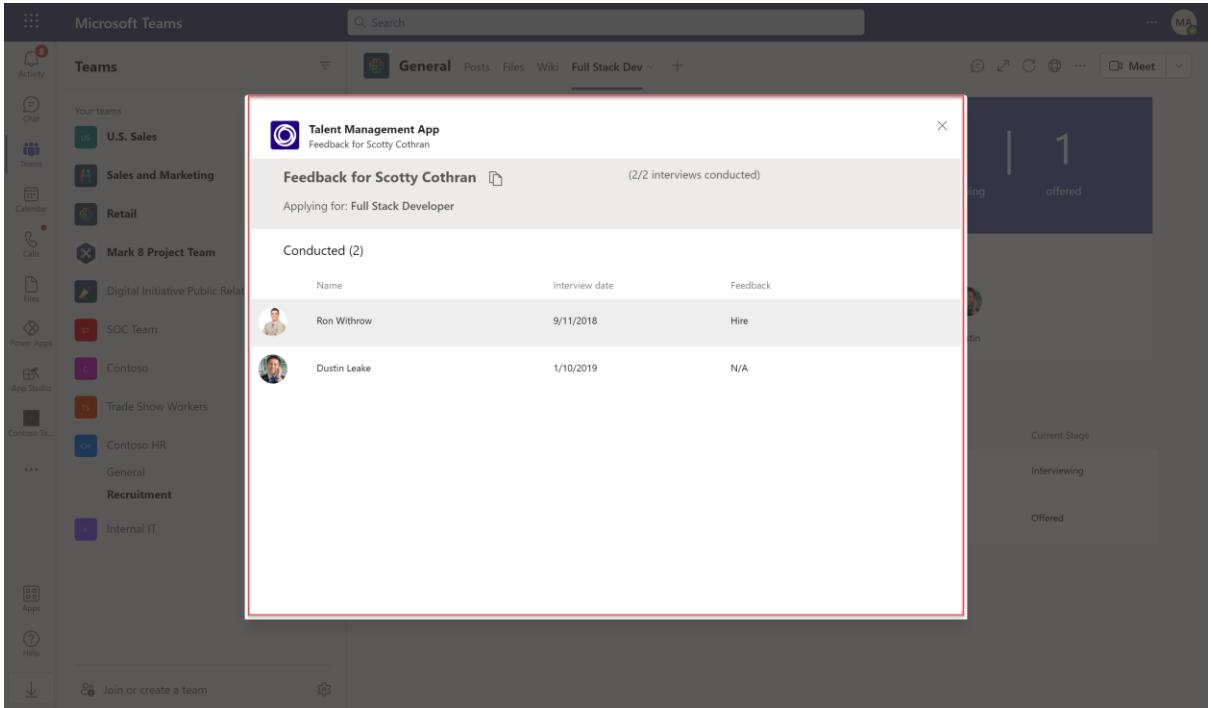
- Callout 1 points to `TeamTab.html`.
- Callout 2 points to `TeamTabConfig.html`.
- Callout 3 points to `TeamTabConfig.html` in the Solution Explorer list.

11. Go back into Teams and open the Team Channel Tab and select one of the candidates that have applied for the Full Stack Developer position.

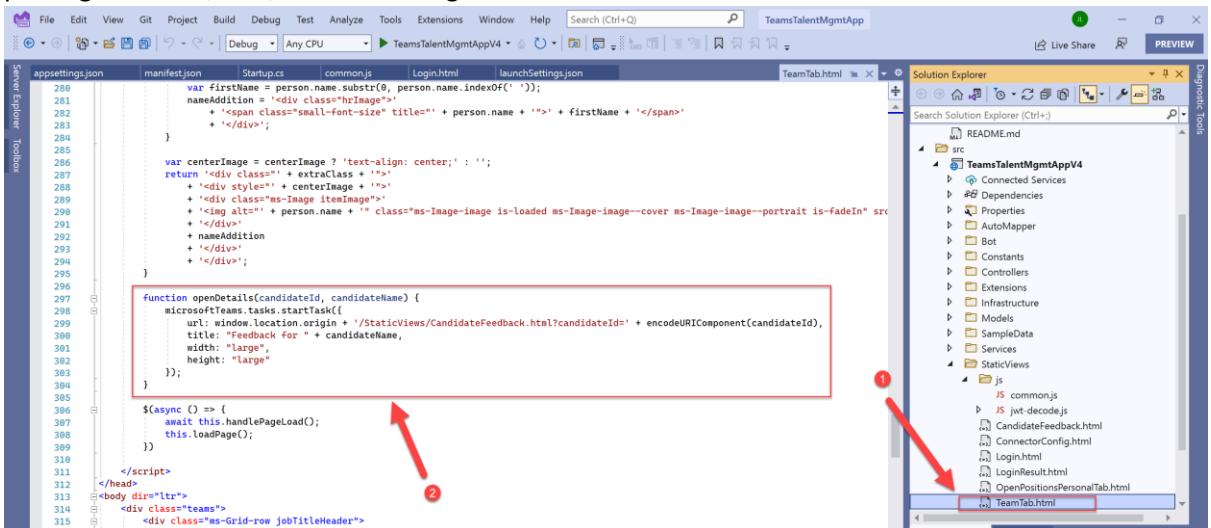
The screenshot shows the Microsoft Teams interface. On the left is the sidebar with various team and app icons. The main area shows a team channel for "U.S. Sales". A job posting for "Full Stack Developer" is displayed, showing the hiring manager is Henry Wickham, ID B6626FOC, and the location is Chicago, IL. Below the job details, under "Applicants", there are two candidates listed: Scotty Cothran and Donna Delaney. Red arrows point from the "Applied" status in the job listing to the "Applied" column in the applicant table, and from the "Offered" status in the job listing to the "Offered" column in the applicant table.

Name	Modified	Location	Current Stage
Scotty Cothran	Yesterday	New York, NY	Interviewing
Donna Delaney	Yesterday	Dallas, TX	Offered

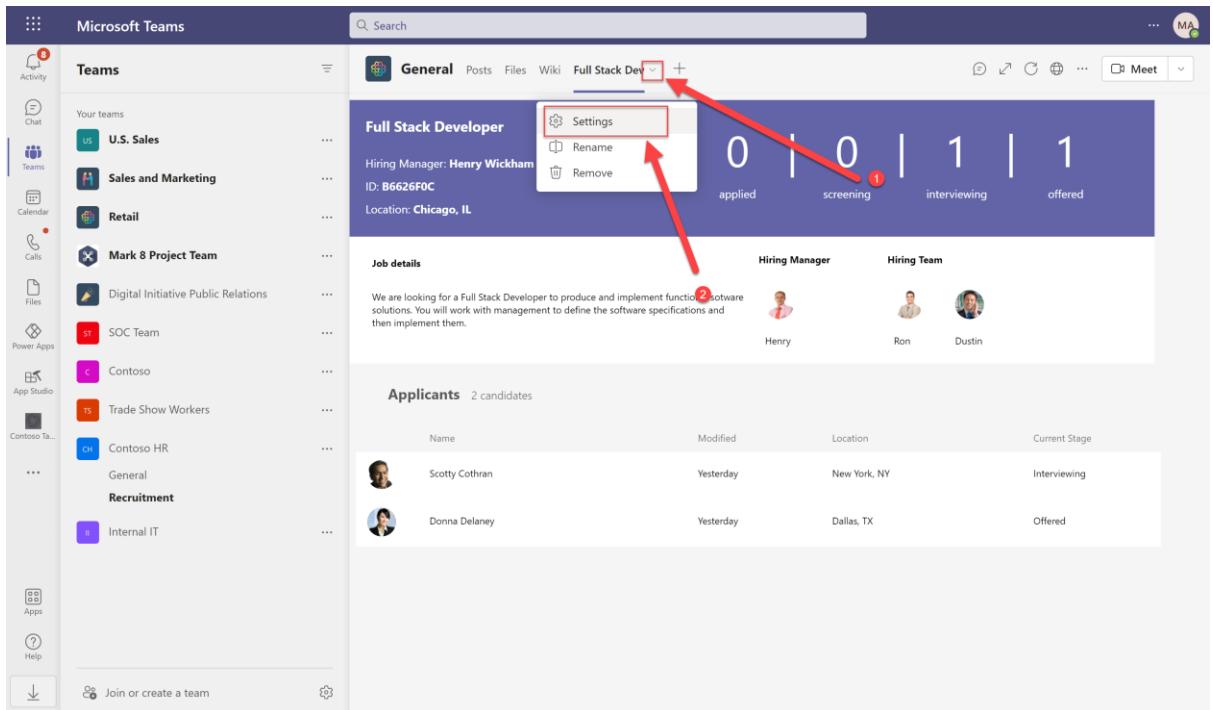
12. This will open a **Task Module**, which is a pop-up within Teams that iFrames a web-application (similar to how Tabs work!). These are great, as they allow you to stay in the context of what you are doing within Teams, while reviewing information that exists within a third party application. Let's review the code that initiated the task module...



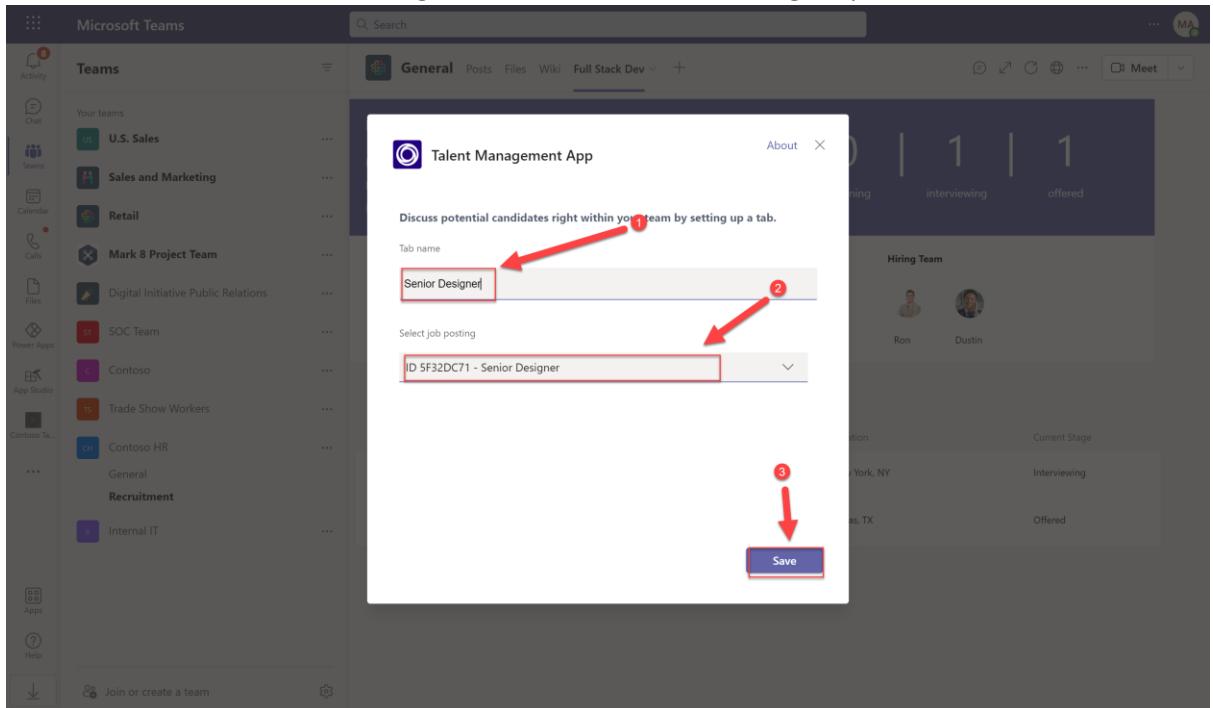
13. Head back into Visual Studio and open the **TeamTab.html** file. Scroll down to line 297 and review the **openDetails** function. This uses the **microsoftTeams.tasks.startTask** function, passing in the url, title, width and height of the task module.



14. Head back into Teams and let's review how you can edit Team Channel Tabs, which will re-invoke the **TeamTabConfig.html** page. Select the **drop-down button** and click **Settings**.



15. Enter the Tab name of Senior Designer and select the Senior Designer position. Click Save



16. You will see now that the TeamTabConfig.html page has changed the entityId to match the positionId of the Senior Designer position. If you want to; review the context in the developer tools to confirm this.

Senior Designer

Hiring Manager: **Mindy Cooper**
ID: **5F32DC71**
Location: **San Francisco, CA**

2 applied | 0 screening | 0 interviewing | 1 offered

Job details

We are looking for a Senior Designer to work with our Product and Engineering team to design new successful products.

Hiring Manager

Mindy

Applicants 3 candidates

Name	Modified	Location	Current Stage
Bart Fredrick	Yesterday	New York, NY	Applied
Wallace Santoro	Yesterday	New York, NY	Offered
Natalia Gill	Yesterday	Miami, FL	Applied

Now that the Channel Tab is working we will now move onto configuring the ChatBot. As we are now moving from HTML and JavaScript to .Net/C#, you will see the following screenshots in dark-mode - a favourite amongst back-end developers...

4) Implement a Personal ChatBot

In this step we will modify the application manifest to add our bot to the app.

1. Open the Developer Portal by navigating to <https://dev.teams.microsoft.com>. Once there we will select **Apps** from the navigation menu and find the application we had previously created.

Developer Portal

+ New app | Import app

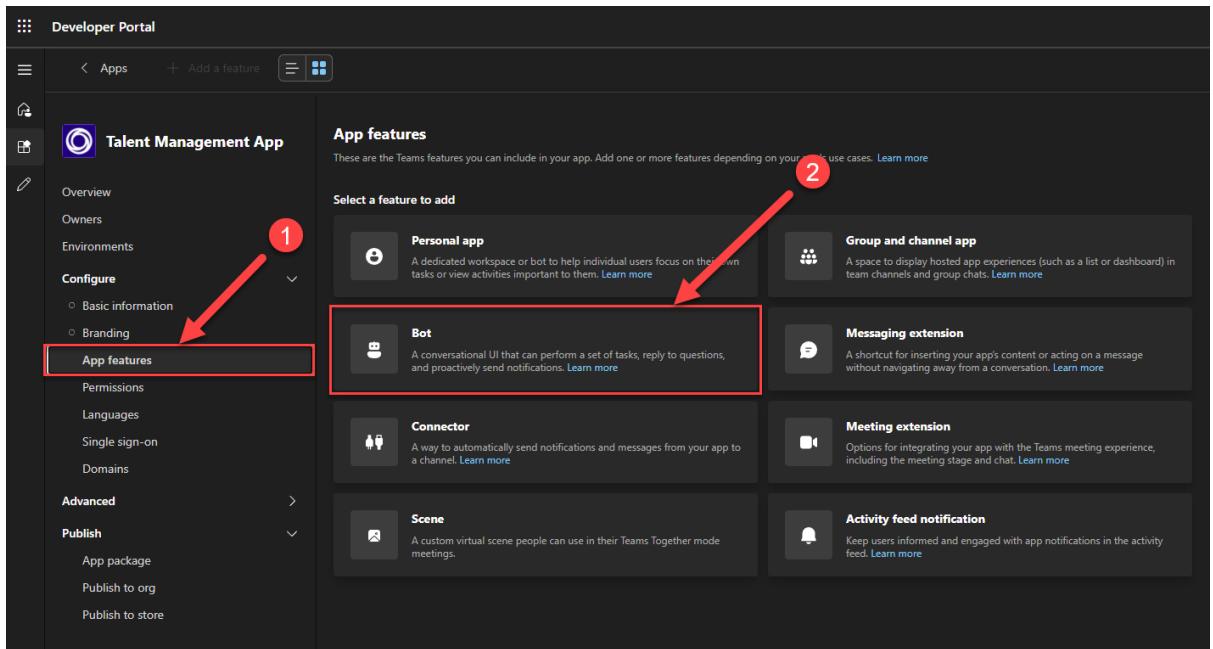
Home | Apps | Tools

Apps

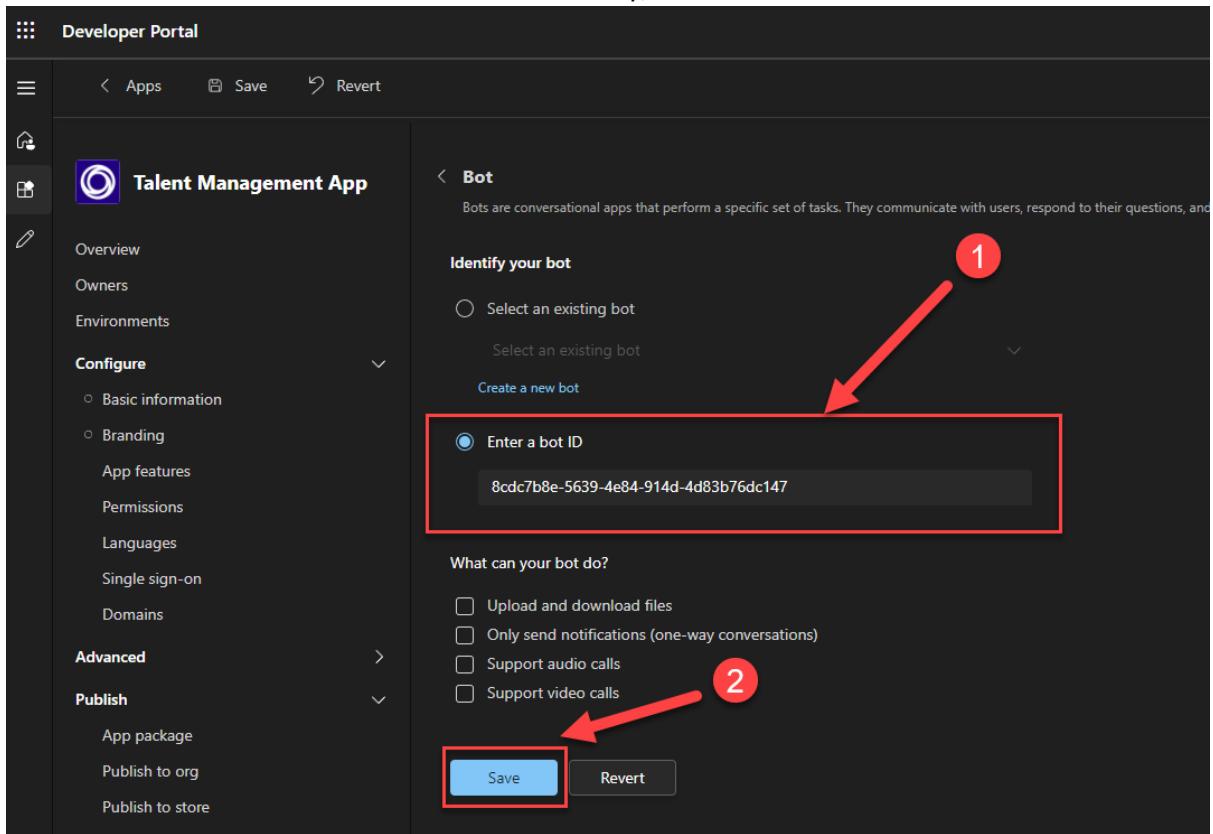
Select an app to see more information or update its configurations.

Name	Version number	App ID	Updated
(app name)	(version)	f5df45ea-0e19-49f3-8cb0-a0d7d9b9aa7e	February 26, 2021
(app name)	(version)	16d25ba6-4c76-45a6-980f-7b16c8559c7	May 27, 2021
DataverseIntegrationSample	1.0.0	40d3c5d1-97a7-4fe2-8e83-91f928684d7	September 13, 2021
Debug Bot	1.0.0	2453049e-c914-c7-b561-1b15a03b333	June 29, 2021
DebugBot2	1.0.0	8d99d45-cff6-4e66-b50d-ba58cf4d5ea	June 30, 2021
DeepLinkSample	1.0.0	c8c75fd2-773e-41f3-861f-73d01885fa1	July 15, 2021
Meeting App Test	1.0.0	4ae1da45-9cca-4752-8bcc-8c9bb4b80cd	June 08, 2021
My Test Bot	1.0.0	a1c38e15-266c-47db-88f0-b1be5da358c6	May 06, 2021
RepliBotApp v2	1.0.0	9redf74a-33fb-48b5-9d57-8ec1c71f9fc3	June 03, 2021
Sample	1.0.0	673f534-0d70-4d1f-8ee-ad34b36131c1	March 04, 2021
Sample App	1.0.0	c368ca89-e0ce-49b8-ab47-e4637f43b93a	August 26, 2021
Talent Management App	1.0.0	82f511dd-e5ef-4370-b7a2-8dc8eff183e3	October 06, 2021
Teams-Sample-TM V2	1.0.0	cea7737f-d317-44d5-9eca-b0a1962c6f02	October 05, 2021
Teams-Sample-TalentManagement	1.0.0	cea7737f-d317-44d5-9eca-b0a1962c6f02	September 30, 2021

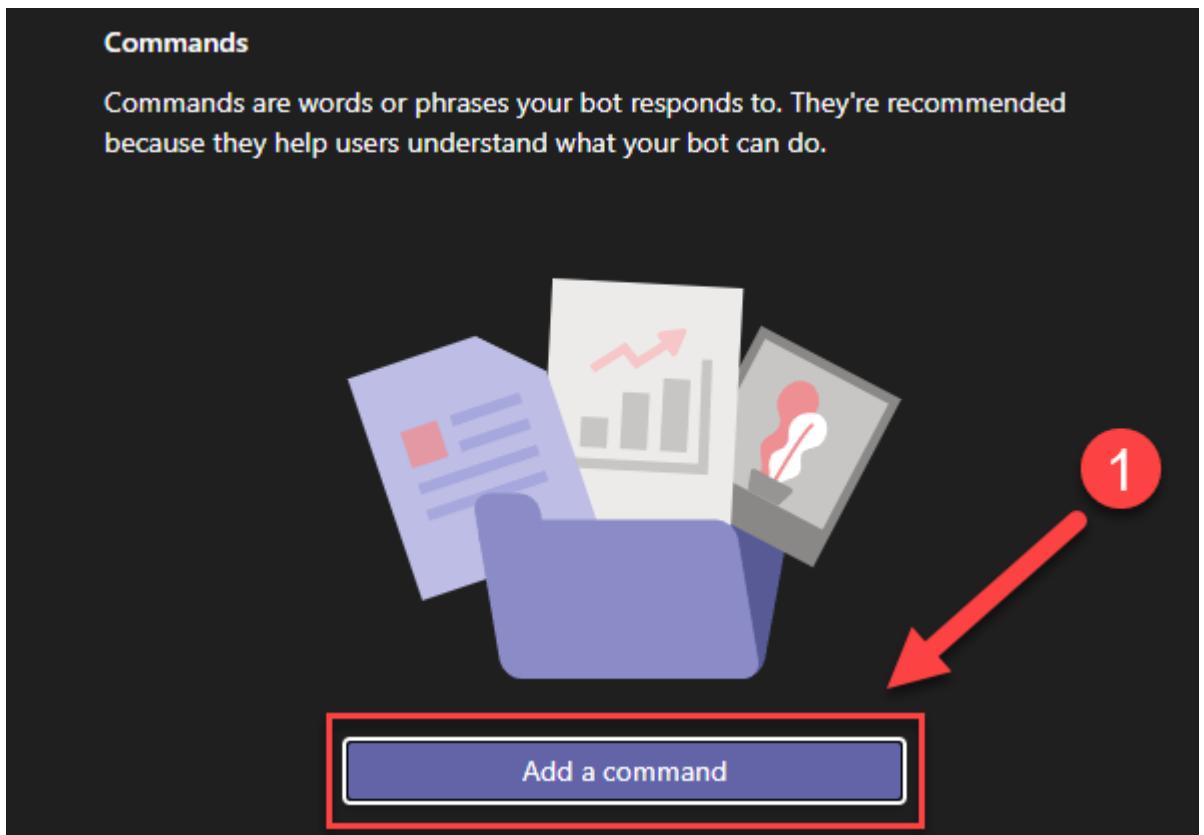
2. Once we are in the context of our application, we will add the bot. Select **App Features** from the left menu and choose **Bot** as the feature we would like to add.



3. Here we will modify the configuration to use the bot service instance we created earlier. There is a bug in the Developer Portal here and adding an existing bot ID doesn't enable the Save button. In order to get around this issue, either **select an existing bot first** (if you have one, or **create a new bot**) Once the dropdown has a bot and the **Save** button has been enabled, click **Enter a bot ID** and copy in the **Client Id** from the App Registration created earlier... the Client Id and bot ID are the same! Finally, click **Save** to save the manifest.



4. Now we want to add commands to the bot, a new Commands dialog should now be visible. Click the **Add a command** button.



- Now we will configure the command. Enter the details below and click **Add**.

Add a bot command

Command*
1 help

Description (help text)*
2 Get a list of commands this bot can accept

Select the scopes in which people can use this command

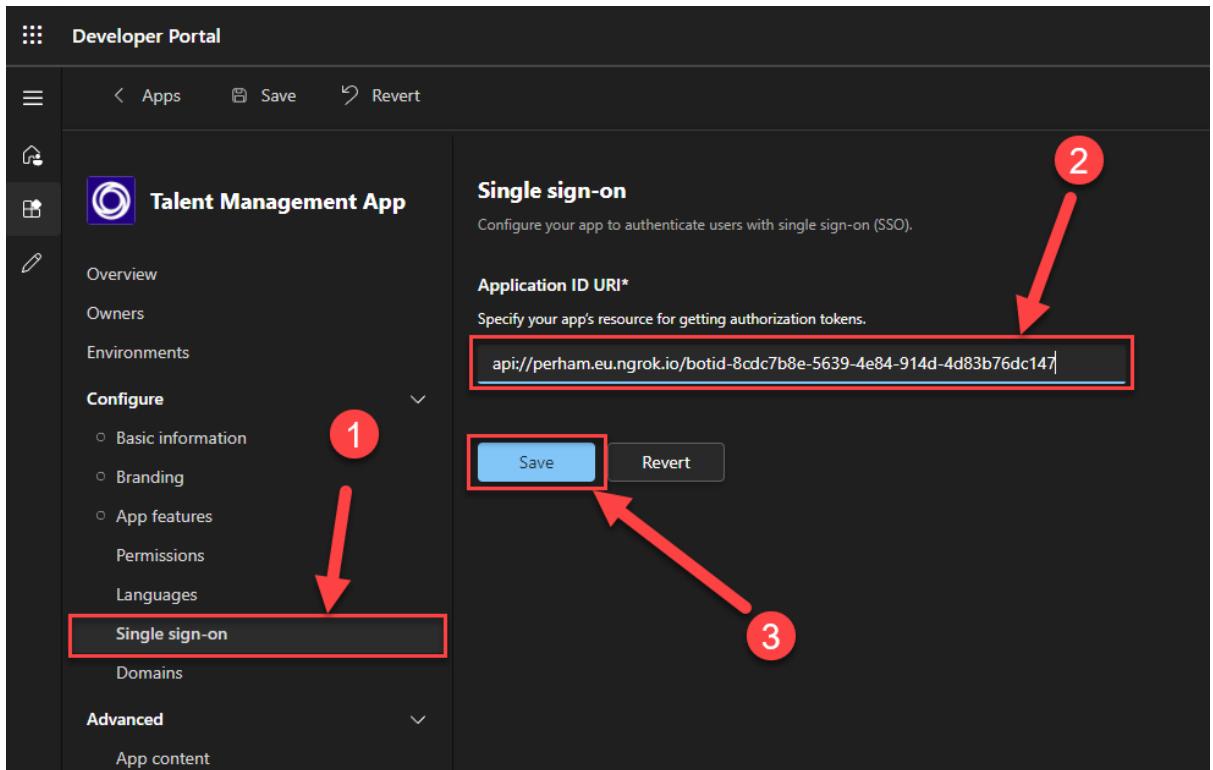
3 Personal
 Team
 Group Chat

4 Add

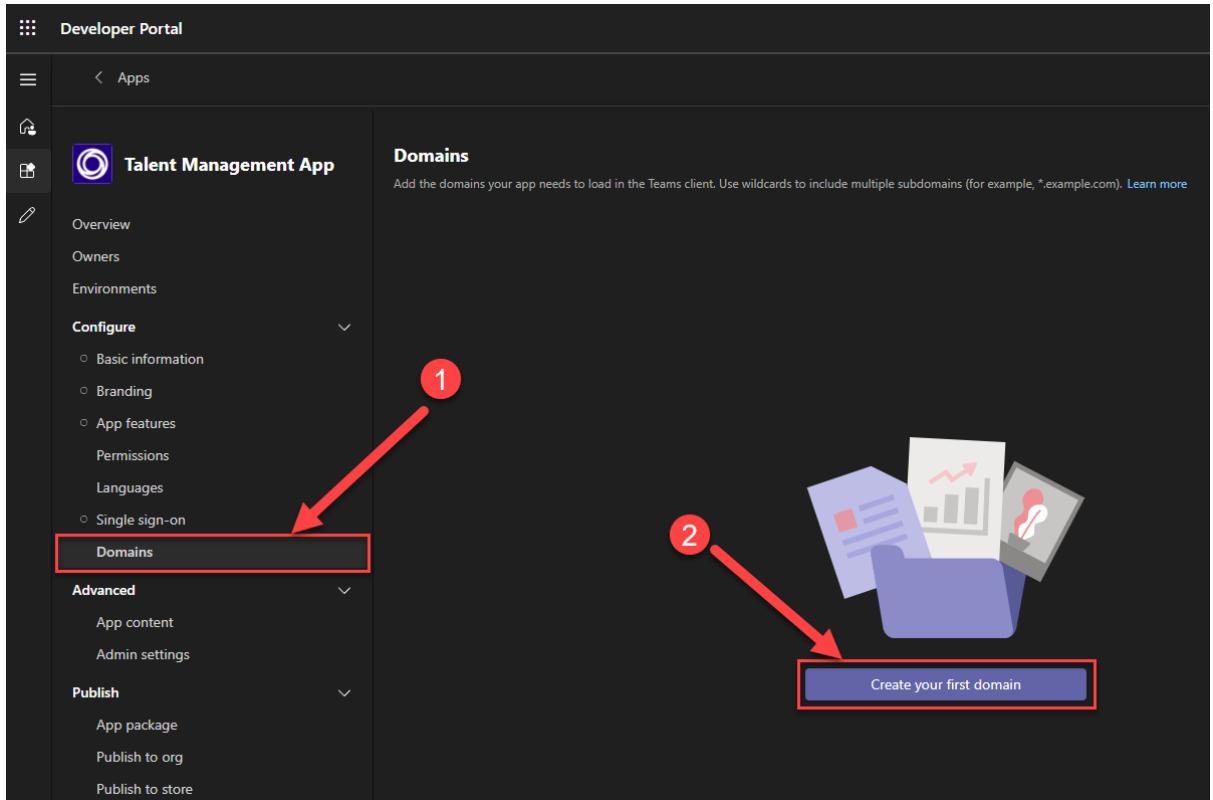
- Repeat this process for all the commands the bot is able to perform as shown below.
NOTE: Ensure that you click **Save** after each command to ensure that you don't lose any configuration if the edit breaks!

Commands			
Command	Description	Scope	
help	Get a list of commands this bot can accept	group, personal, team	...
candidate details	Show details about a candidate, for example: candidate details Bart Fredrick	group, personal, team	...
summary	Show summary about a candidate, for example: summary Bart Fredrick	group, personal, team	...
top candidates	Show top candidates for a Position ID, for example: top candidates 10CD3166	group, personal, team	...
new job posting	Create a new job posting	group, personal, team	...
open positions	List all your open positions	group, personal, team	...
+ Add a command			
<input style="background-color: #0078D4; color: white; padding: 5px; margin-right: 10px; border-radius: 5px; border: 1px solid #0078D4; font-weight: bold; font-size: 14px; width: 150px; height: 40px;" type="button" value="Save"/> <input style="background-color: #E0E0E0; color: #0078D4; padding: 5px; border-radius: 5px; border: 1px solid #0078D4; font-weight: bold; font-size: 14px; width: 150px; height: 40px;" type="button" value="Revert"/>			

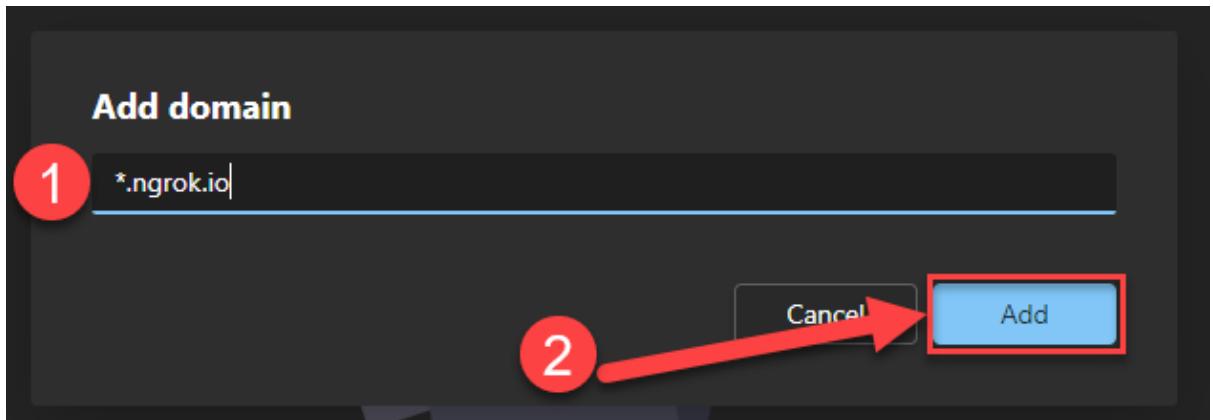
7. Next will configure Single sign-on so that your bot can work in the context of the current user. Select **Single sign-on** from the left menu. Enter your Application ID URI as defined in your App Registration earlier (this should be in the following format: `api://{{domain}}/botid-{{client id}}`). Finally, click **Save**.



- Now that the bot has been configured and SSO has been enabled, we need to allow the bot to talk to both the Azure bot service and also your backend API service. Select **Domains** from the left menu and then click **Create your first domain**.



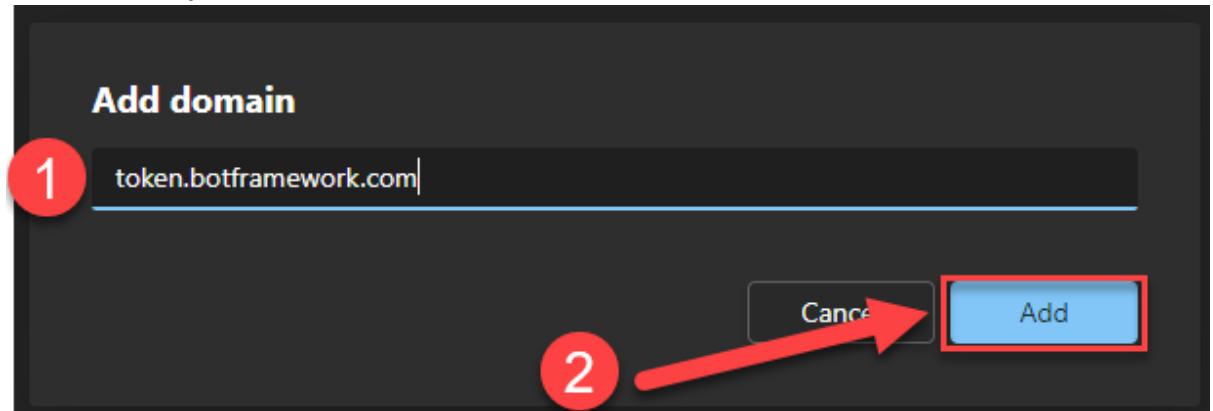
- There are two domains we will need to add, the first will be your API service. As we are using ngrok to tunnel, add the wildcard domain `*.ngrok.io` and the click **Save**.



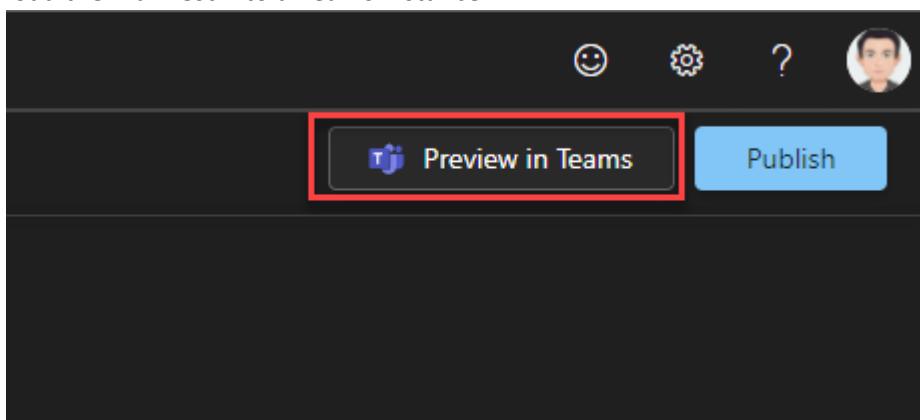
10. You will now see that your domain appears in the domains list. Click **Add a domain** and repeat this process.

A screenshot of a 'Domains' list page. It shows a table with two columns: 'Domains' and 'Source'. There is one entry: *.ngrok.io with Source 'Additional'. Below the table is a red-bordered button labeled '+ Add a domain'.

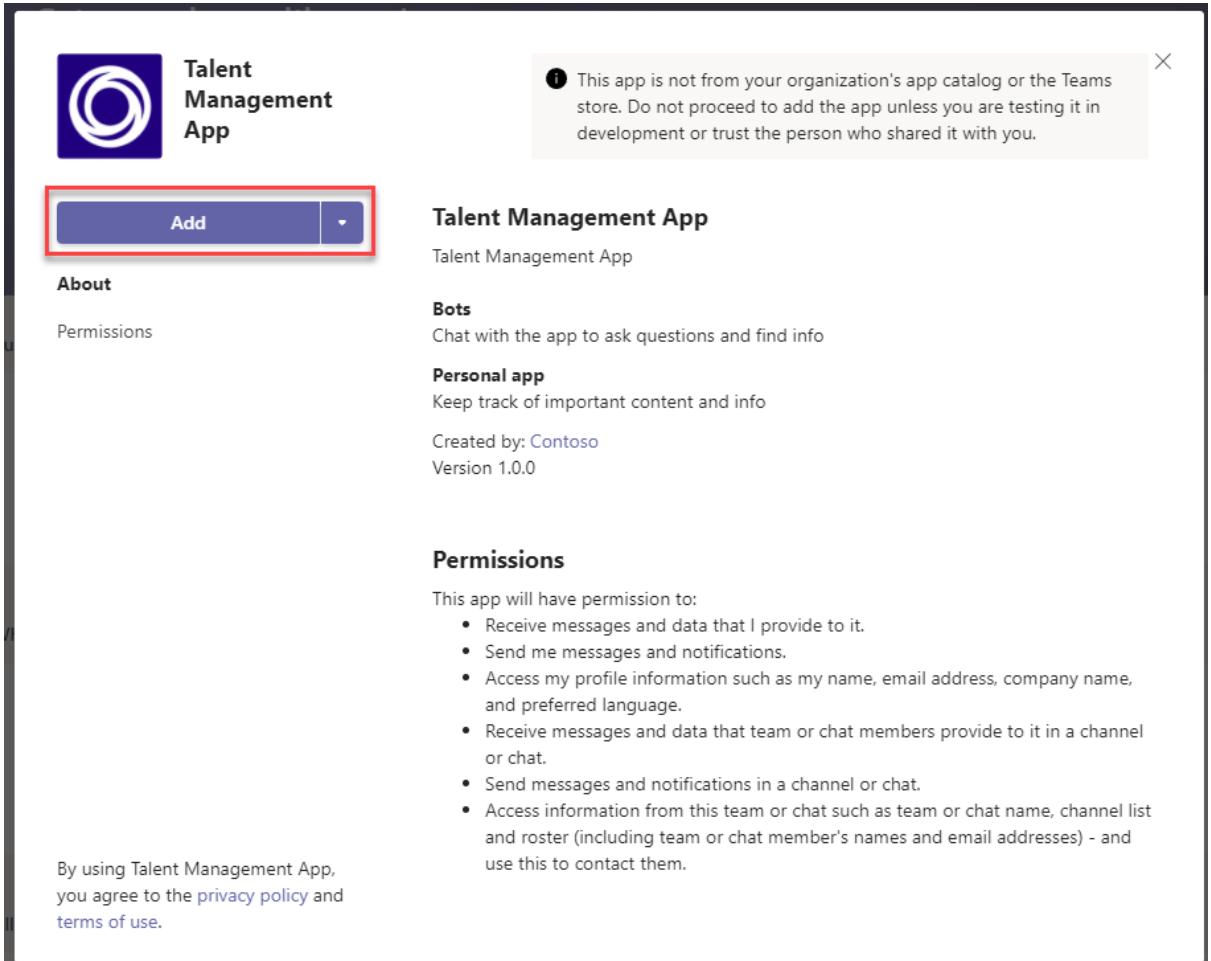
11. We will now add the domain that bot framework uses to community with our bot for SSO. Enter *token.botframework.com* and click **Add**.



12. Now that the bot has been configured completely, let's test it out! Click **Preview in Teams** to load the manifest into a Teams instance.

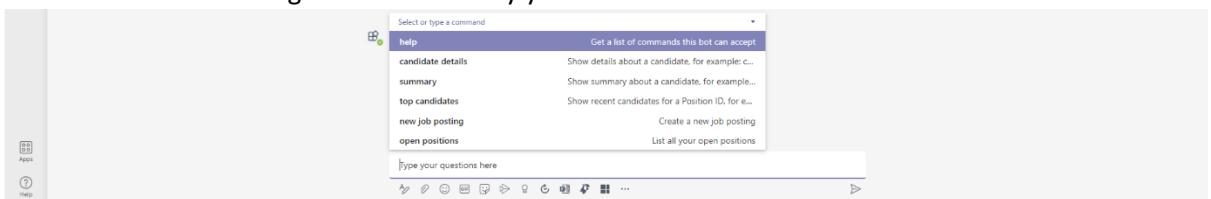


13. Once in Teams, you should be displayed with the following dialog asking you to add your application. This will actually overwrite the manifest that already exists and you will see that the permissions list has grown now that you've added your bot. Click **Add** to update your application and be taken to the channel.

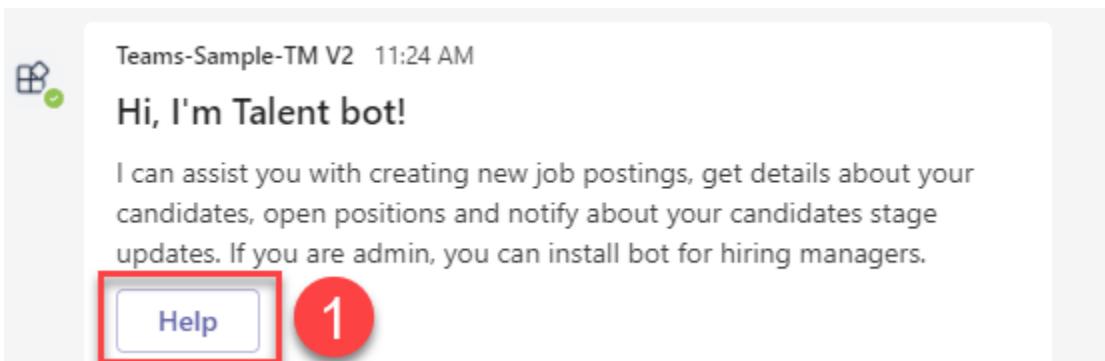


14. When you highlight the Compose box, you will now see that you are presented with a list of command available through the bot. You can either click a command or type the name.

Note: that the bot will have also added a welcome message but depending on the size of your window it might be hidden by the bot command list. Once the list is dismissed you will see the welcome message that was sent by your bot.



15. This message includes an action button which has been configured send the help command to the bot. Clicking Help will send the text "help" to the bot just as if you'd typed it yourself in the compose box.



16. Once the command has been received by the bot it will respond with a friendly help message listing some of the common commands.

Teams-Sample-TM V2 11:24 AM

Hi, I'm Talent bot!

I can assist you with creating new job postings, get details about your candidates, open positions and notify about your candidates stage updates. If you are admin, you can install bot for hiring managers.

Help

11:26 AM help

Teams-Sample-TM V2 11:26 AM

Here's what I can help you do:

- Show details about a candidate, for example: candidate details Bart Fredrick
- Show summary about a candidate, for example: summary Bart Fredrick
- Show top recent candidates for a Position ID, for example: top candidates EDCDAF3F
- Create a new job posting
- List all your open positions

Type your questions here

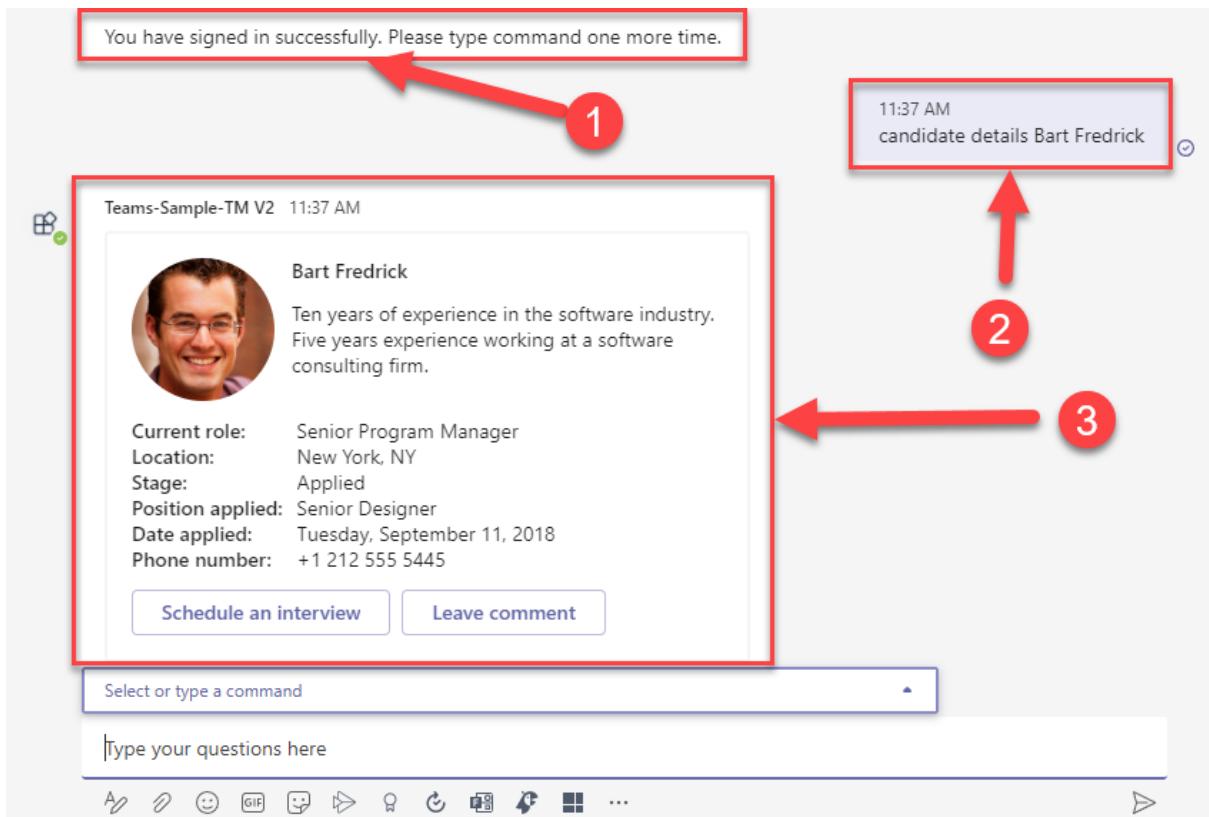
...

17. One such command is the candidate details command. Let's try this now... copy or type *candidate details Bart Fredrick* to see details about the candidate.

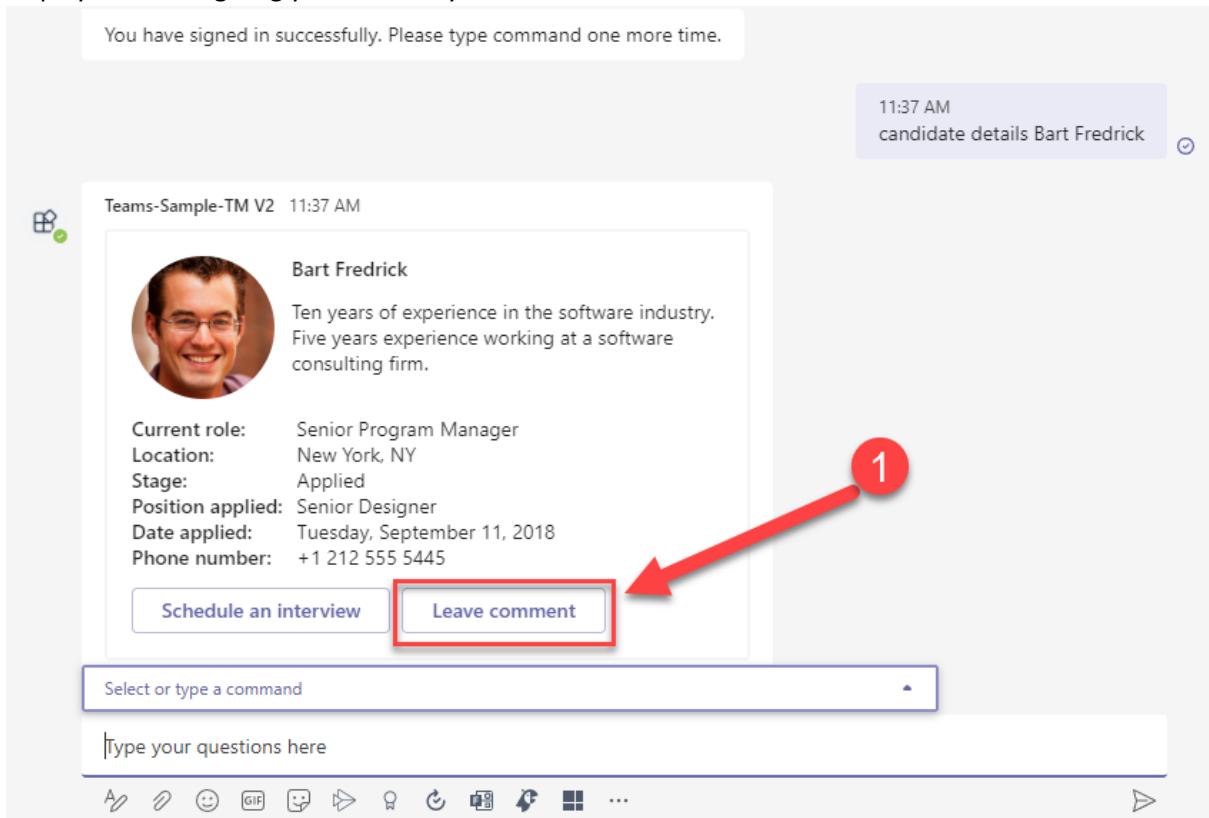
candidate details Bart Fredrick

...

18. You will notice that before any candidate details are shown you are presented with a message saying that you have successfully been signed in. This is the single sign-on process working in the background. The message will also say that now you are signed in you must resend your command. Doing this will display an Adaptive Card displaying details about Bart Fredrick.



19. You will also notice that there are some buttons on the card. Clicking **Leave comment** will display a textbox giving you the ability to leave a comment about the candidate.



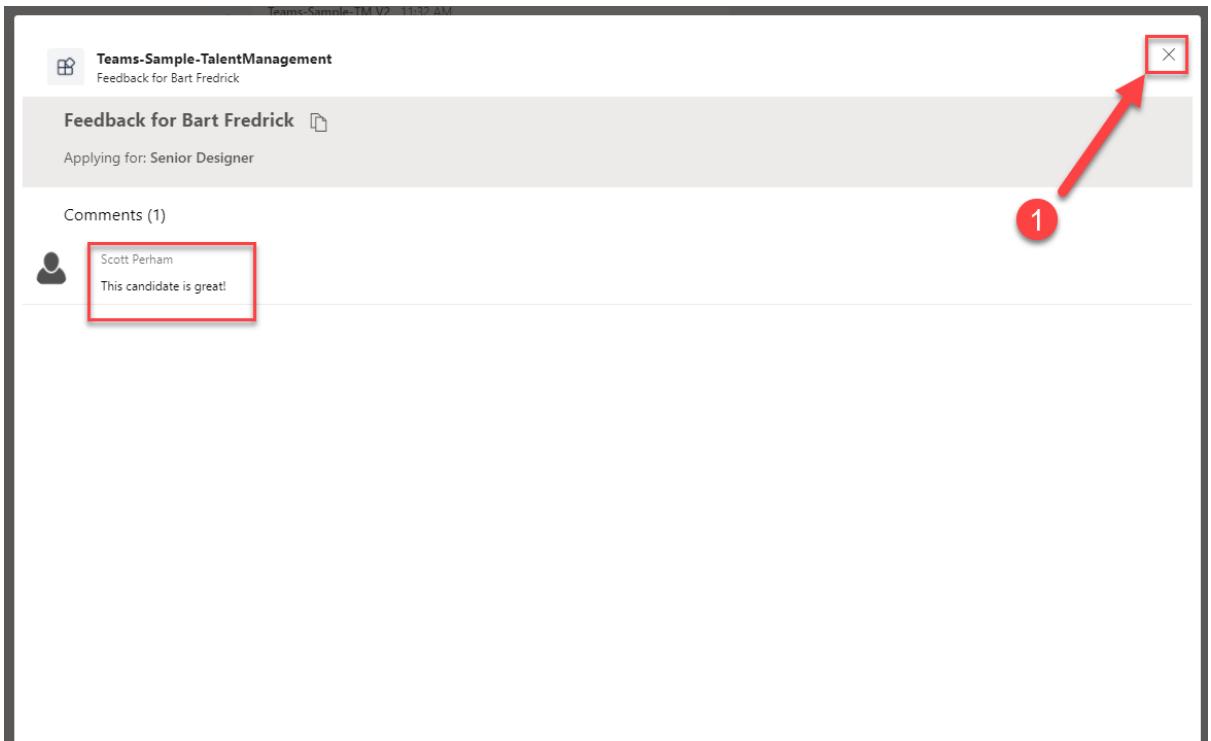
20. Enter a comment and click **Submit** to send the comment to the bot.

The screenshot shows the Microsoft Teams Task Module for a candidate named Bart Fredrick. At the top, there's a profile picture of Bart, his name, and a brief bio: "Ten years of experience in the software industry. Five years experience working at a software consulting firm." Below this, there are several details about the application: Current role: Senior Program Manager; Location: New York, NY; Stage: Applied; Position applied: Senior Designer; Date applied: Tuesday, September 11, 2018; Phone number: +1 212 555 5445. There are two buttons: "Schedule an interview" and "Leave comment". A red box highlights a comment "This candidate is great!" and a red arrow labeled "1" points to it. Another red box highlights the "Submit" button, and a red arrow labeled "2" points to it. At the bottom, there's a text input field "Type your questions here" with various emoji and icon buttons above it.

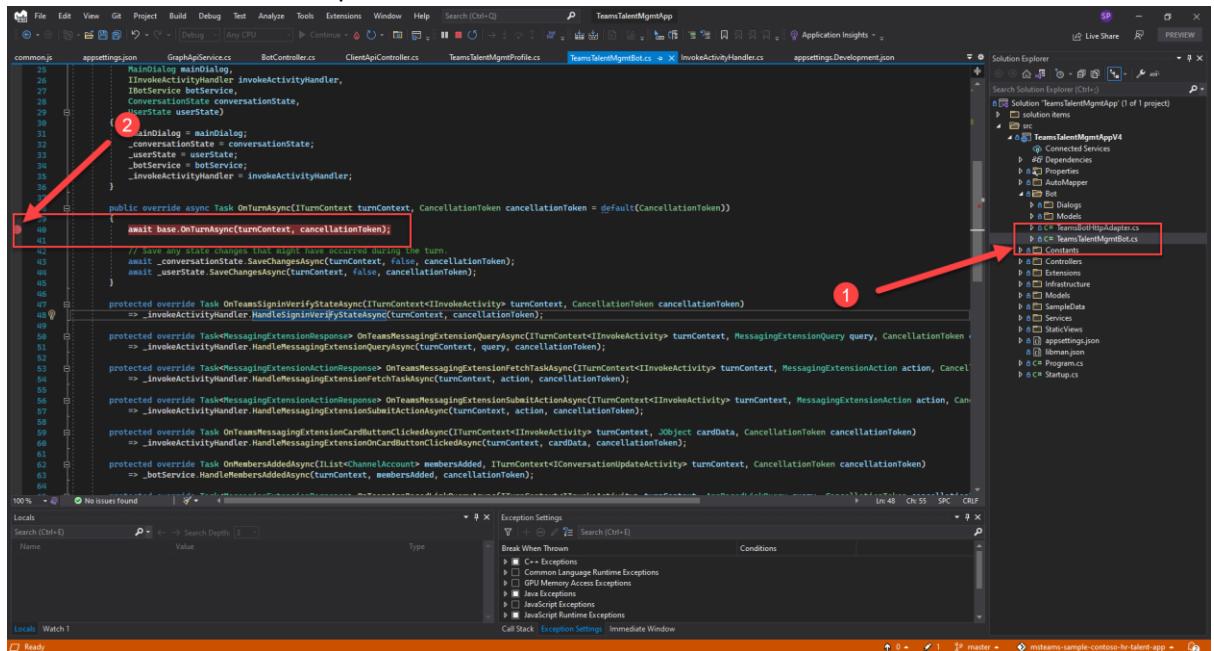
21. This will resend the candidate adaptive card to Teams and because we now have some feedback, and additional (conditional) button will be shown. Click **Open candidate feedback** to see a summary of all the feedback left for the candidate.

The screenshot shows the Microsoft Teams Task Module for the same candidate, Bart Fredrick. The profile, bio, and application details are identical to the previous screenshot. The "Leave comment" button has been replaced by a new button labeled "Open candidate feedback", which is highlighted with a red box and a red arrow labeled "1". The "Schedule an interview" button remains. The bottom section is identical to the previous screenshot, featuring a text input field "Type your questions here" and a row of emoji and icon buttons.

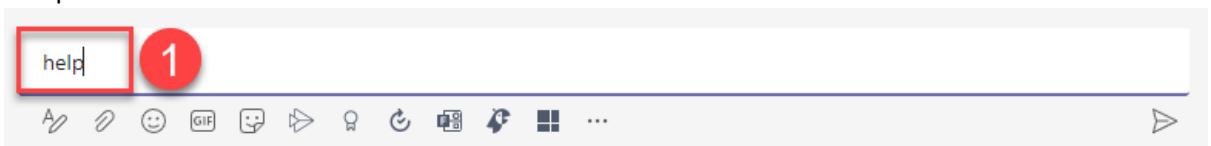
22. You will see your feedback in the list! Once you are done click the X in the corner to close the Task Module.



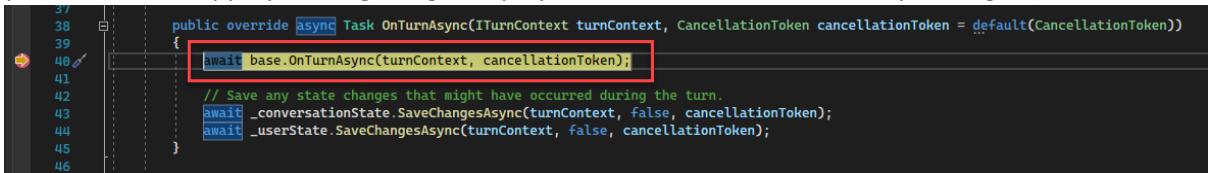
23. We will now have a look behind the scenes and see what happens when commands are sent from Teams to your bot. Open Visual Studio and navigate to the TeamsTalentMgmtBot.cs file. Then put a breakpoint on line 40. (Breakpoint can either be set by clicking the leftmost gutter on the appropriate line, or by clicking somewhere in the line and hitting F9). The line will turn red once the breakpoint has been set.



24. Now with your application running, go back to Teams, in your bot chat channel, type "help" and press *Enter* or the send button.



25. You should see that Visual Studio has halted execution and highlighted line 40 in yellow. This means that this is the current line of execution but the breakpoint has caused the application to pause. (Note that while you have halted execution the bot will not work until you resume the app by clicking the green play button from the toolbar - or pressing F5).



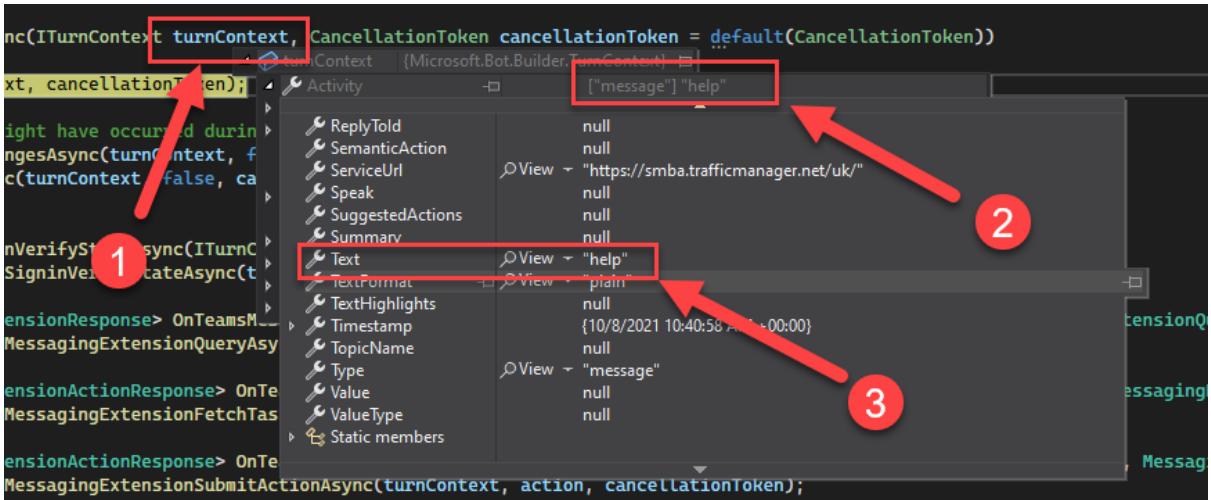
```

37
38     public override async Task OnTurnAsync(ITurnContext turnContext, CancellationToken cancellationToken = default(CancellationToken))
39 {
40     await base.OnTurnAsync(turnContext, cancellationToken);
41
42     // Save any state changes that might have occurred during the turn.
43     await _conversationState.SaveChangesAsync(turnContext, false, cancellationToken);
44     await _userState.SaveChangesAsync(turnContext, false, cancellationToken);
45 }
46

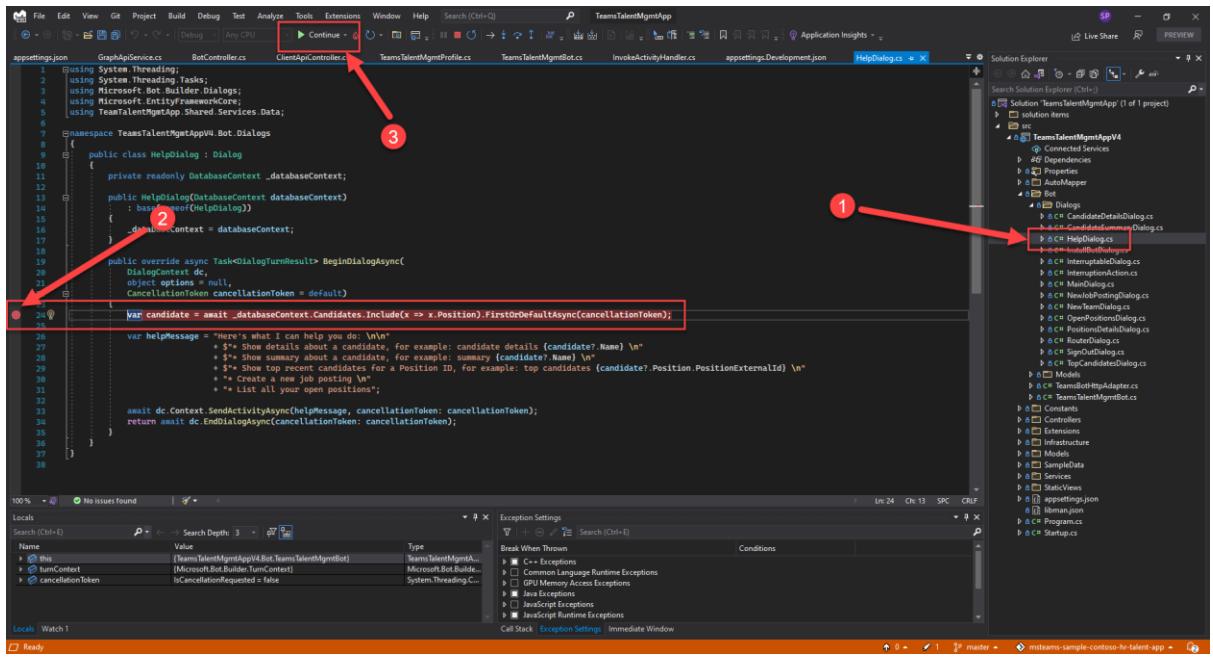
```

26. While we are paused Visual Studio allows us to inspect the value of any variable in the current scope. As we are in the scope of a method called **OnTurnContext**, we can inspect the parameters that were passed to it. To do this, hover your mouse over the parameter called *turnContext* then expand the various treeview items in the pop-out to see the values. In the case of the image below, we can see the text that was passed in the message.

Note: In bot framework, conversational concepts are used to describe the various parts of bot interaction. In the same way that two humans might take turns speaking in a conversation, a Turn in bot framework represents your bots chance to respond to an event.



27. Another concept in bot framework is that of Dialogs. A dialog represents a part of a conversation. This could be a single text response or a conversation that spans multiple turns. It allows us to encapsulate the functionality of a particular conversation. One such dialog is the HelpDialog. Open the HelpDialog.cs file and set a breakpoint on line 24. This click the green play button to resume the bots execution.



28. You will see that your breakpoint has been hit! As before you can inspect the various variable values if you wish.

Finally, this method will send the **helpMessage** back to Teams.

Note: If an execution takes more than a few seconds, Teams will ignore the response as it will assume that the bot has broken and is not responding. You may therefore find that when you resume execution from this point the help message may not actually appear in Teams. This is expected! If you remove both of the breakpoints and send the *help* command again you will see that the help message is displayed as expected.

```

18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
  public override async Task<DialogTurnResult> BeginDialogAsync(
    DialogContext dc,
    object options = null,
    CancellationToken cancellationToken = default)
  {
    var candidate = await _databaseContext.Candidates.Include(x => x.Position).FirstOrDefaultAsync(cancellationToken);
    ...
  }

```

29. There are two more areas of interest in the code that you may be interested to see. The first is the code for SSO. Most of this is handled for us in a dialog provided by the framework, but we still have to handle the users security token once it's been returned to us. This is particularly useful if you want to cache the token for use in the future, perhaps exchange it for a graph token to be able to call graph on the users behalf?

To see how this sample has dealt with the token, open the **InvokeActivityHandler.cs** file and inspect the **HandleSignInVerifyStateAsync** method.

```

public async Task<InvokeResponse> HandleSignInVerifyStateAsync(ITurnContext<ITurnActivity> turnContext, CancellationToken cancellationToken)
{
    var token = ((TurnContext<TurnActivity>)(turnContext.Activity)).Value<string>("token");
    if (token != null && !string.IsNullOrEmpty(token))
    {
        await _tokenProvider.SetTokenAsync(token, turnContext, cancellationToken);
        await turnContext.SendActivityAsync("You have signed in successfully. Please type command one more time.", cancellationToken);
    }
    await _conversationState.ClearStateAsync(turnContext, cancellationToken);
    return new InvokeResponse { Status = (int) HttpStatusCode.OK };
}

```

30. Another area of interest is how the bot responds with adaptive cards. In the CandidateTemplate.cs file is the code used to create a relatively complex card. The image below shows how the bot is configuring the candidate feedback task module to be displayed when **Open candidate feedback** is clicked.

```

if (candidate.Comments.Any() || candidate.Interviews.Any())
{
    var contentUrl = data.AppSettings.BaseUrl + $"/StaticViews/CandidateFeedback.html?candidateId={candidate.CandidateId}";
    card.Actions.Add(new AdaptiveOpenAction
    {
        Title = "Open candidate feedback",
        Url = new Uri(string.Format(
            Constants.TenantTemplateFormat,
            data.AppSettings.MicrosoftAppId,
            Uri.EscapeDataString(contentUrl),
            Uri.EscapeDataString(candidate.Name),
            data.AppSettings.MicrosoftAppId,
            "large",
            "large")));
    });
    var leaveCommentCommand = new Command
    {
        commandId = AppCommands.LeaveInternalComment,
        candidateId = candidate.CandidateId
    };
    var wrapAction = new CardAction
    {
        Title = "Submit",
        Value = leaveCommentCommand
    };
}

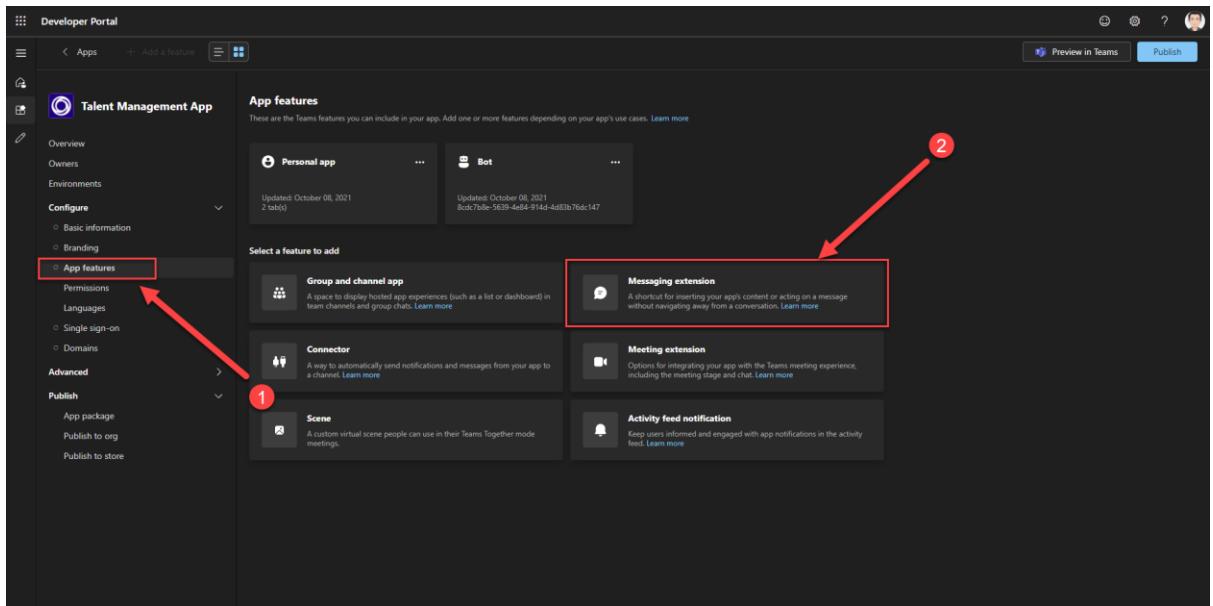
```

Now that the Tab is working...

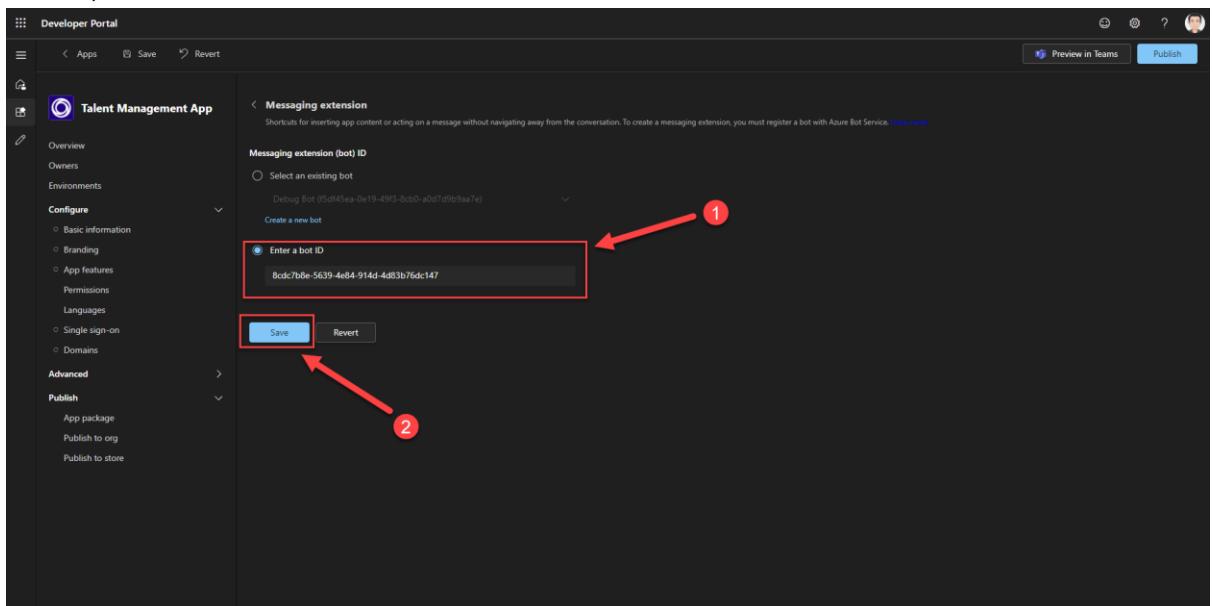
5) Implement a Message Extension Search

In this step we will extend the functionality of Teams to include a search function. Searching for candidates and positions using a message extension

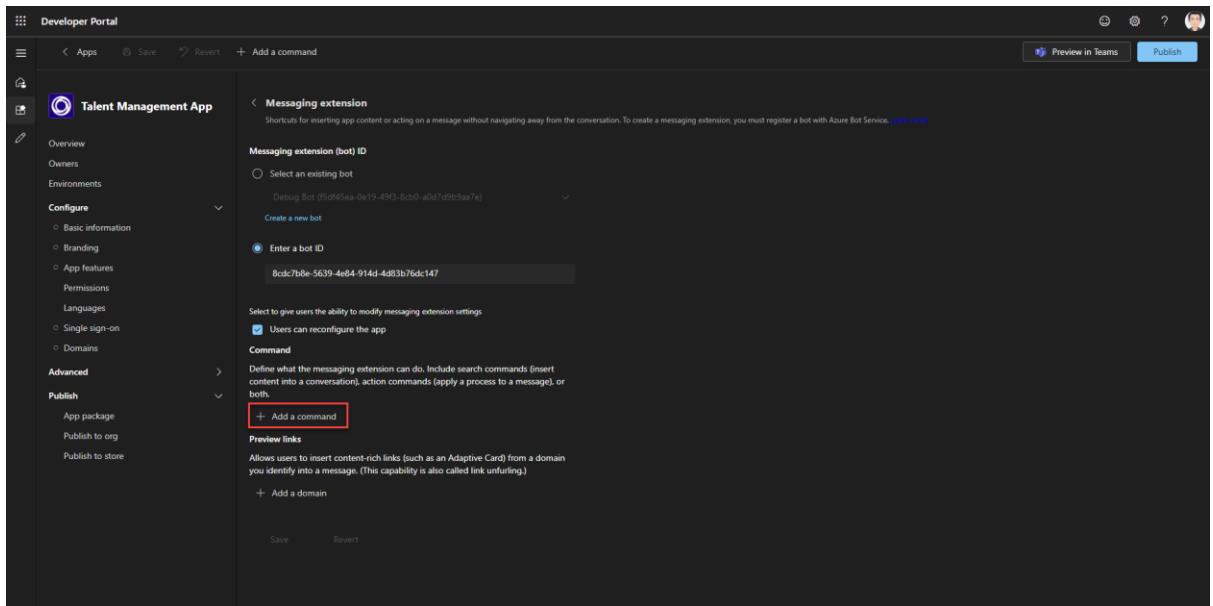
- As with everything else in the manifest we need to go back to the Developer Portal to add a new feature. Select **App Features** from the left menu and click **Messaging extensions**.



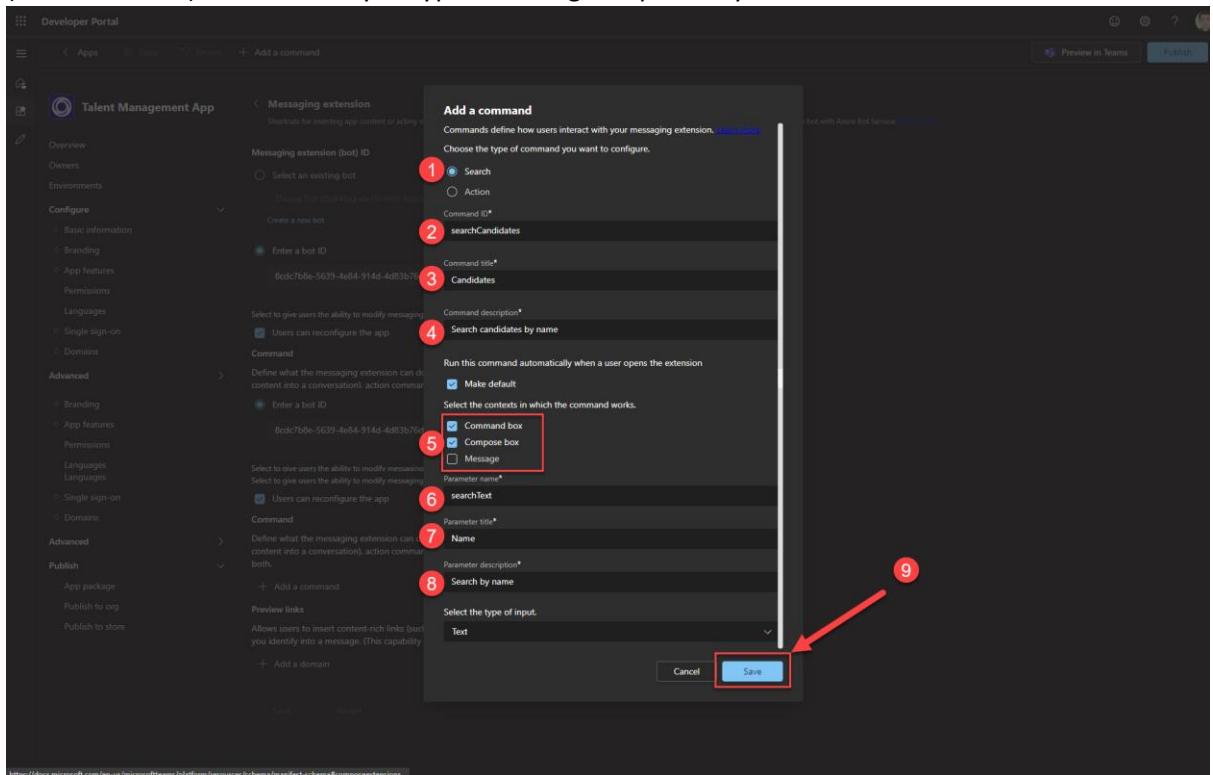
2. Select **Enter a bot id** and enter your App Registration *Client Id* into the textbox. (This is so that you can have a different bot handling your messaging extensions if you wish.) Once this is done, click **Save**.



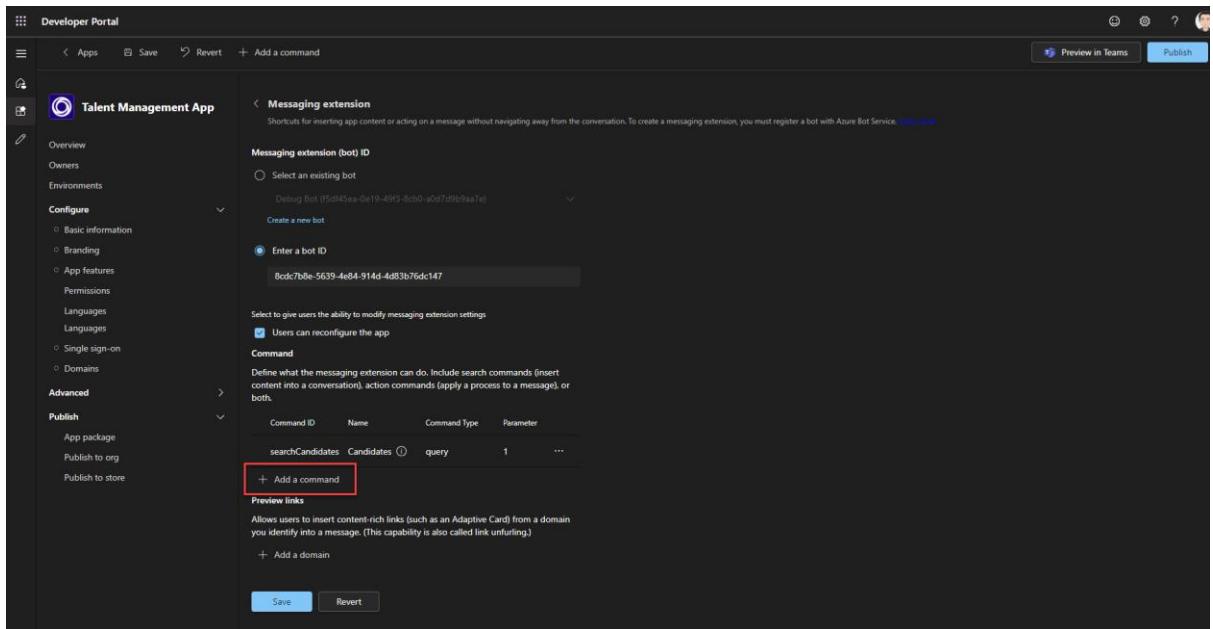
3. Now we need to add the commands. Select **Add a command**.



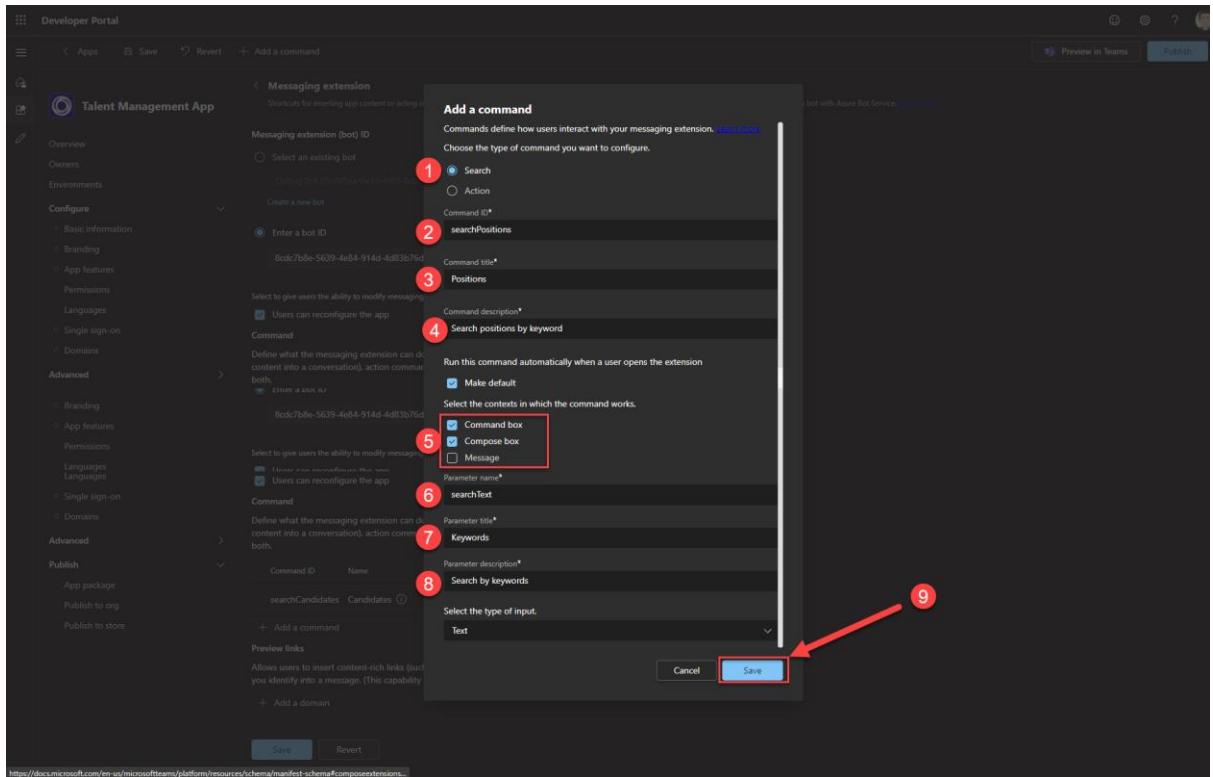
- Enter the values below to add the search candidates search command and click **Save**.
- Note:** we are selecting Command box and Compose box as the areas within Teams that this function will be available. This is the large search box at the top of the Teams interface (Command box) and the area you type a message respectively.



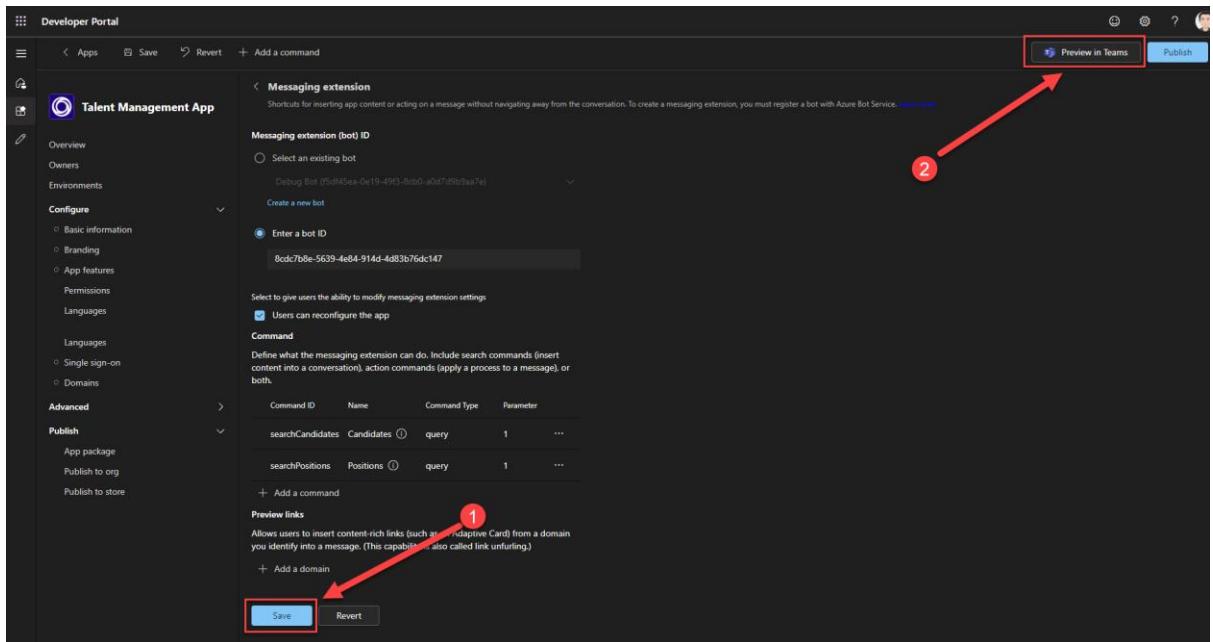
- Once done, we will add a second search command, this one is for the position search function. Click **Add a command** again.



6. Enter the values below and click **Save**.



7. Now you can see the two commands have been added, click **Save** and then **Preview in Teams** to test it out.



- As before you will be prompted to “add” your app and as before this will overwrite the existing manifest inside Teams.

Talent Management App

This app is not from your organization's app catalog or the Teams store. Do not proceed to add the app unless you are testing it in development or trust the person who shared it with you.

Add

About

Talent Management App

Bots

Chat with the app to ask questions and find info

Messages

Insert content from the app directly into messages

Personal app

Keep track of important content and info

Created by: Contoso
Version 1.0.0

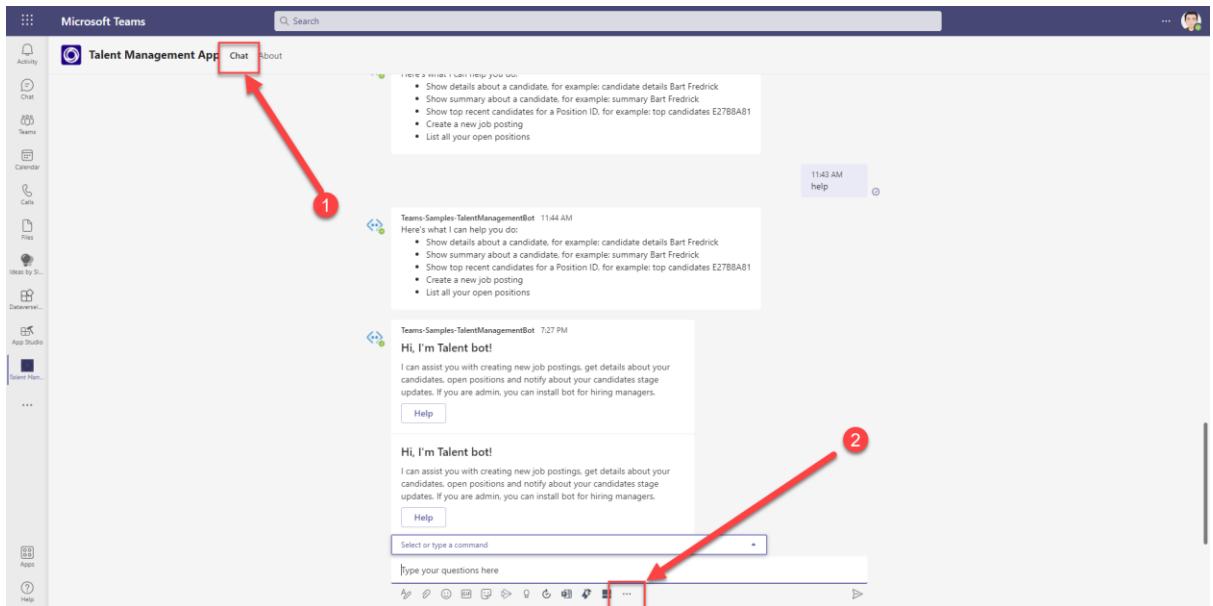
Permissions

This app will have permission to:

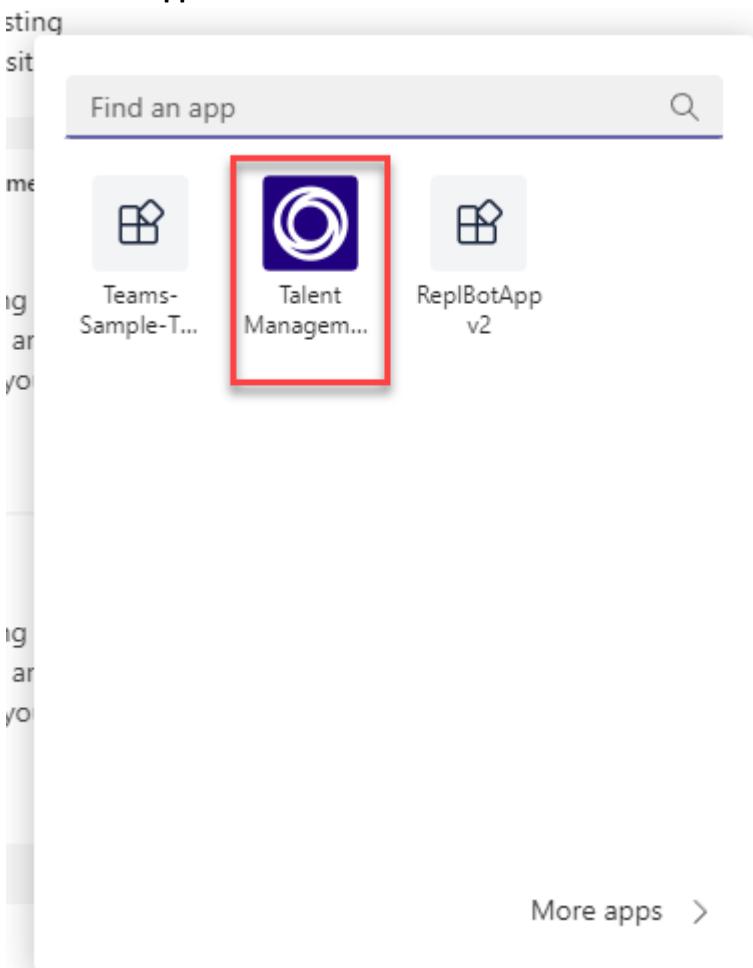
- Receive messages and data that I provide to it.
- Send me messages and notifications.
- Access my profile information such as my name, email address, company name, and preferred language.
- Receive messages and data that team or chat members provide to it in a channel or chat.
- Send messages and notifications in a channel or chat.
- Access information from this team or chat such as team or chat name, channel list and roster (including team or chat member's names and email addresses) - and

By using Talent Management App, you agree to the [privacy policy](#) and [terms of use](#).

- Ensure that you are in the Chat tab for your application. (If you have completed this document in order you will see other tabs here, but **Chat** is where we interact with our bot). Then click the ellipsis (...) next to the icons below the compose box.



10. Within the dialog that's displayed you should see your application listed, if not you can use the **Find an app** textbox to search for it.



11. Selecting your application will display the following dialog. The things to note here are that there are two search functions **Candidates** and **Positions** shown as tabs below the **Search by name** textbox. Select one of the candidates.

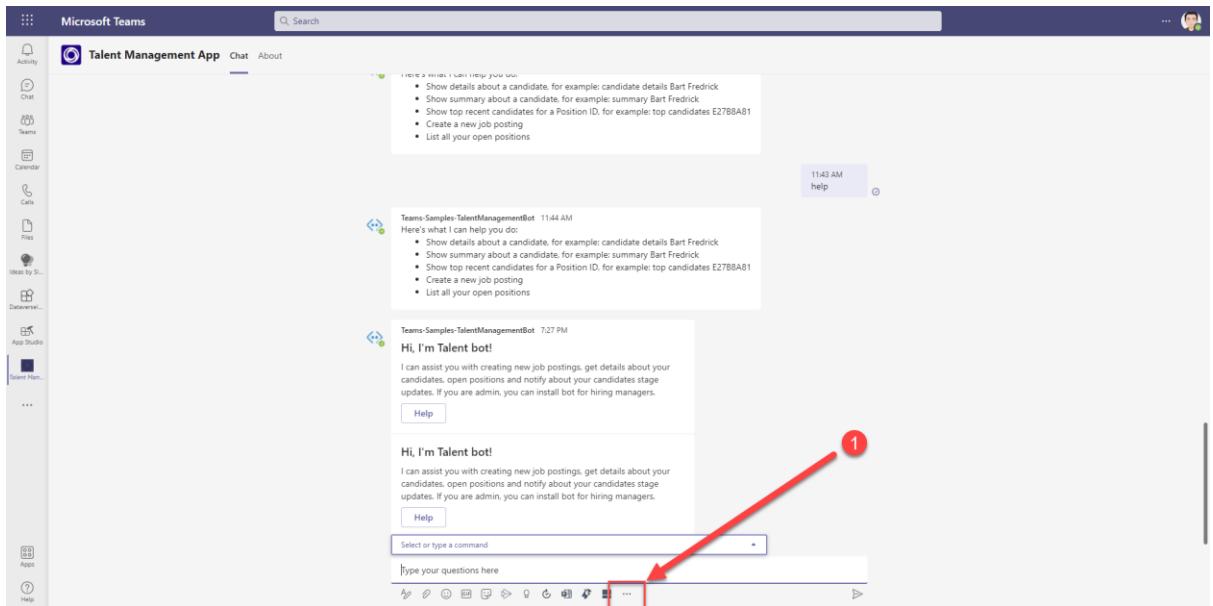
The screenshot shows a web-based application titled "Talent Management App". At the top, there is a search bar labeled "Search by name" with a magnifying glass icon. Below the search bar are two tabs: "Candidates" (which is underlined in blue) and "Positions". The main content area displays a list of five candidates, each with a small profile picture and their name and current role. The first candidate, "Bart Fredrick" (Senior Program Manager | New York, NY), is highlighted with a red rectangular box.

- Bart Fredrick
Current role: Senior Program Manager | New York, NY
- Scotty Cothran
Current role: Software Developer II | New York, NY
- Donna Delaney
Current role: Software Developer II | Dallas, TX
- Wallace Santoro
Current role: Marketing Manager | New York, NY
- Samuel Rodman
Current role: Software Developer II | San Francisco, CA

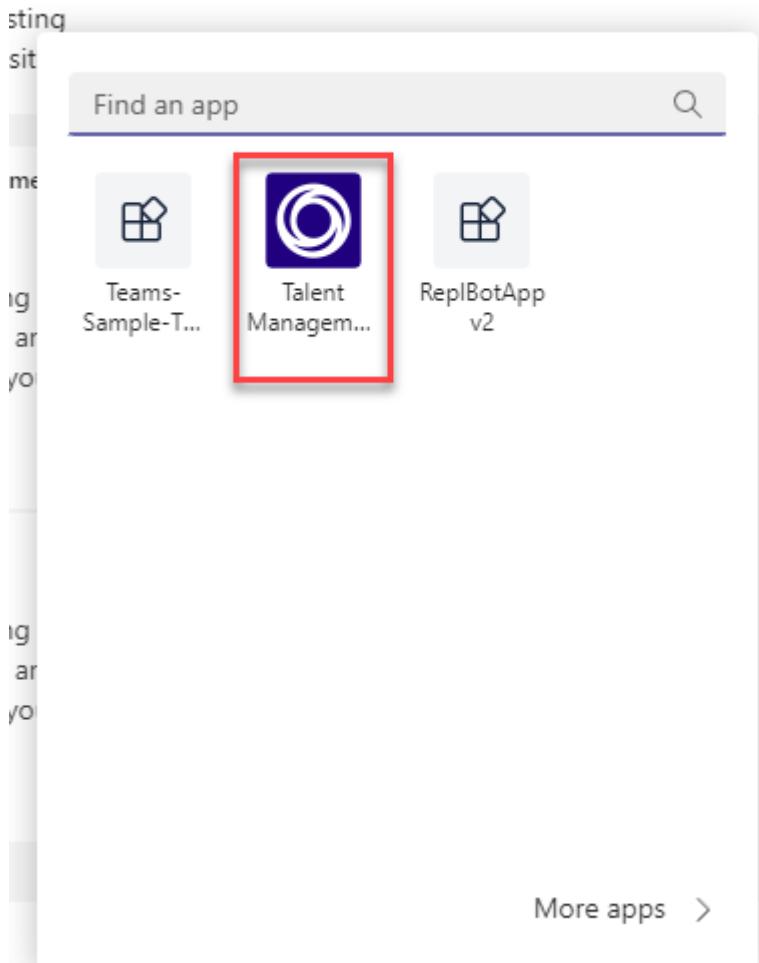
12. This will place the adaptive card we saw earlier in the compose box. It's important to note that message extensions don't post directly to the chat, instead they add their content to the compose box and allow you to include more text before choosing to send the content.

The screenshot shows a Microsoft Teams chat window. On the left, there is a sidebar with various icons for Activity, Chat, Teams, Calendar, Calls, Files, Idea by S... (disabled), Database, App Studio, and Talent M... (selected). The main area shows a conversation between "Teams-Samples/TalentManagementBot" and the user. The bot has sent a message with a link to "List all your open positions". Below this, the user has sent a message: "Hi, I'm Talent bot!". The bot has responded with its description: "I can assist you with creating new job postings, get details about your candidates, open positions and notify about your candidates stage updates. If you are admin, you can install bot for hiring managers." A help button is also present. At the bottom, there is an adaptive card for "Bart Fredrick" with a red border. The card includes a profile picture, the candidate's name, and detailed information: "Ten years of experience in the software industry. Five years experience working at a software consulting firm." It also lists the current role (Senior Program Manager), location (New York, NY), stage (Applied), position applied (Senior Designer), date applied (Tuesday, September 11, 2018), and phone number (+1 212 555 5445). There are buttons for "Schedule an interview" and "Leave comment".

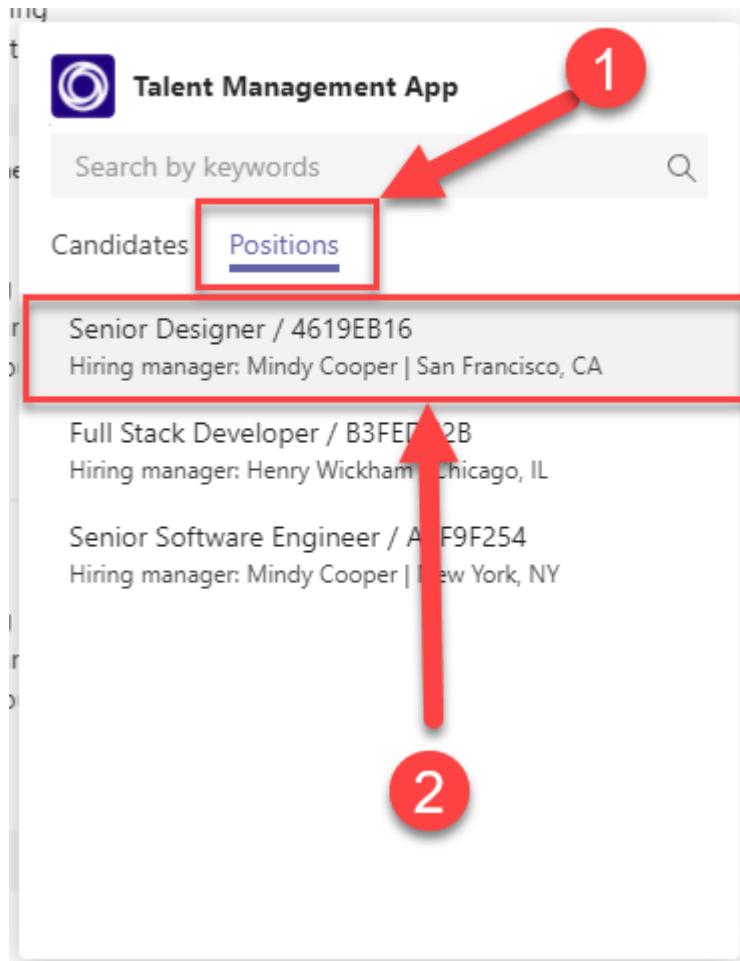
13. Repeat this process to see the positions adaptive card. Click the ellipsis again (...).



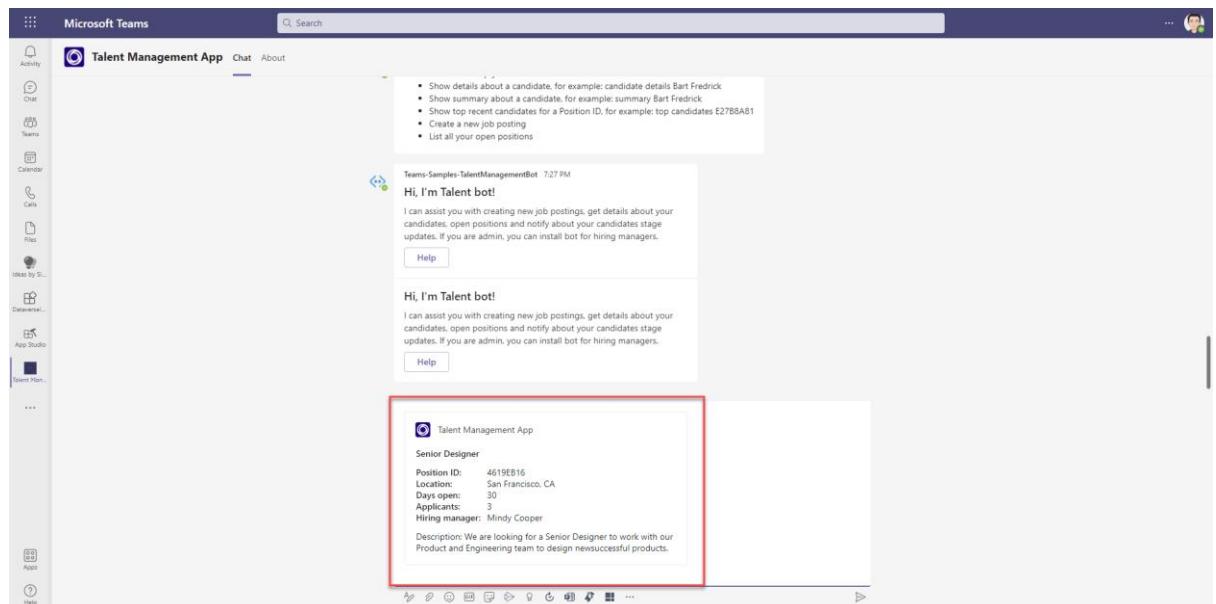
14. Find your application



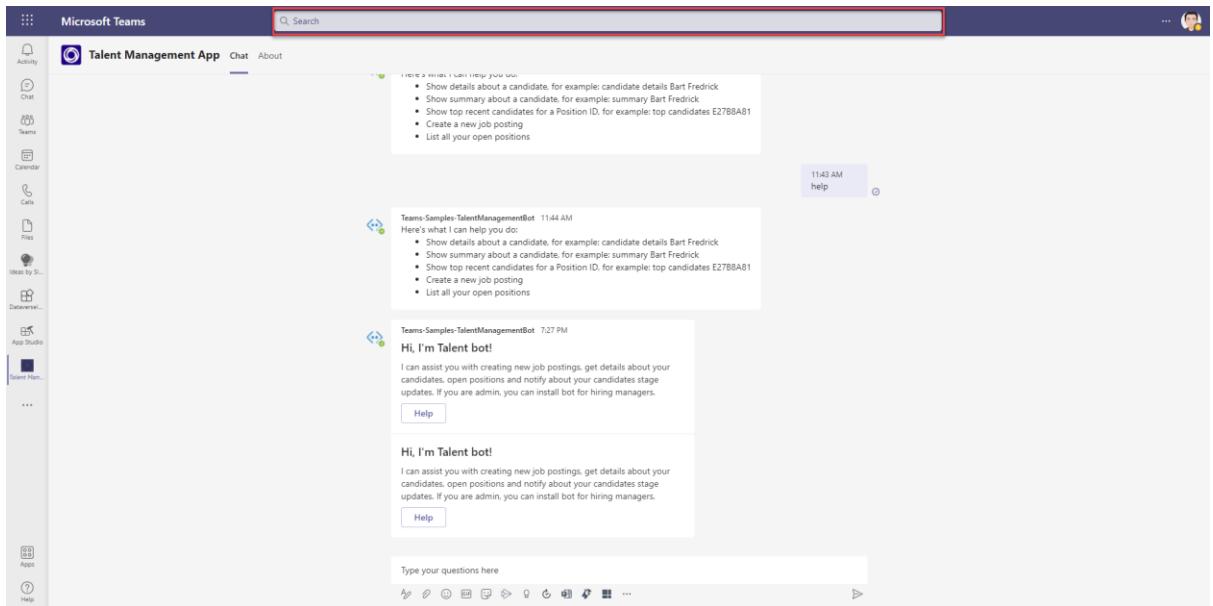
15. This time, select **Positions**. You will see a list of positions displayed. Now select one.



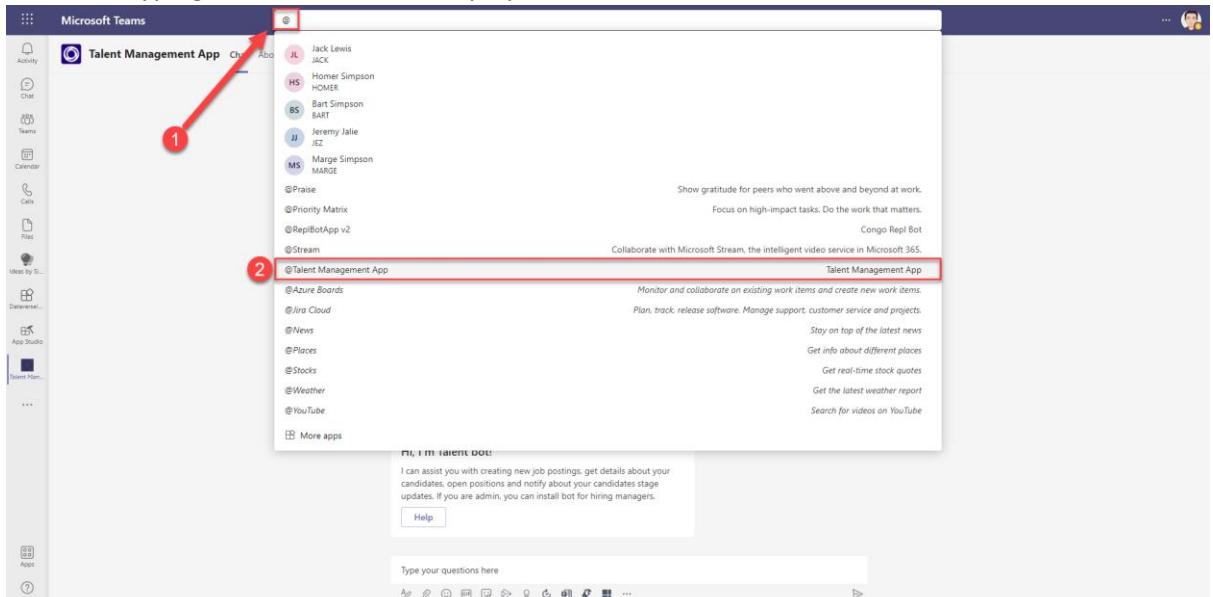
16. As before you will see that an adaptive card has been added to the compose box ready to be sent!



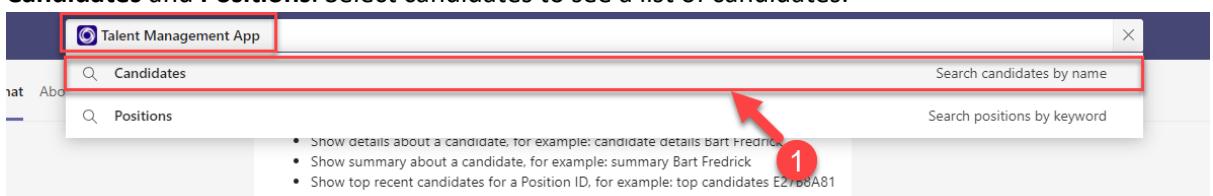
17. Another feature of a search command is that it can be made available to the global search bar (Command box) at the top of the Teams interface.



18. Typing an @ symbol will show a dropdown with a list of installed bots. If you can't see yours then start typing the name until it's displayed and then select it from the list.



19. This will change the context of the search box to your bot only and once in that context you will be prompted with all the available search functionality exposed by our bot. In this case **Candidates and Positions**. Select candidates to see a list of candidates.



20. The search box has changed context again as is now only working within the Candidates search function. You can search for a candidate by name, or select one from the list.

Talent Management App > Candidates		[Search by name]
	Bart Fredrick Current role: Senior Program Manager New York, NY	
	Scotty Cothran Current role: Software Developer II New York, NY	
	Donna Delaney Current role: Software Developer II Dallas, TX	
	Wallace Santoro Current role: Marketing Manager New York, NY	
	Samuel Rodman Current role: Software Developer II San Francisco, CA	

21. Unlike the compose message extension, the adaptive card is displayed directly in the search box dropdown, but works like any other adaptive card.

 Talent Management App > Candidates Bart Fredrick X



Bart Fredrick
Ten years of experience in the software industry.
Five years experience working at a software consulting firm.

Current role: Senior Program Manager
Location: New York, NY
Stage: Applied
Position applied: Senior Designer
Date applied: Tuesday, September 11, 2018
Phone number: +1 212 555 5445

[Schedule an interview](#) [Leave comment](#)

22. To give you an idea of how this works, let's open Visual Studio and inspect some variables!

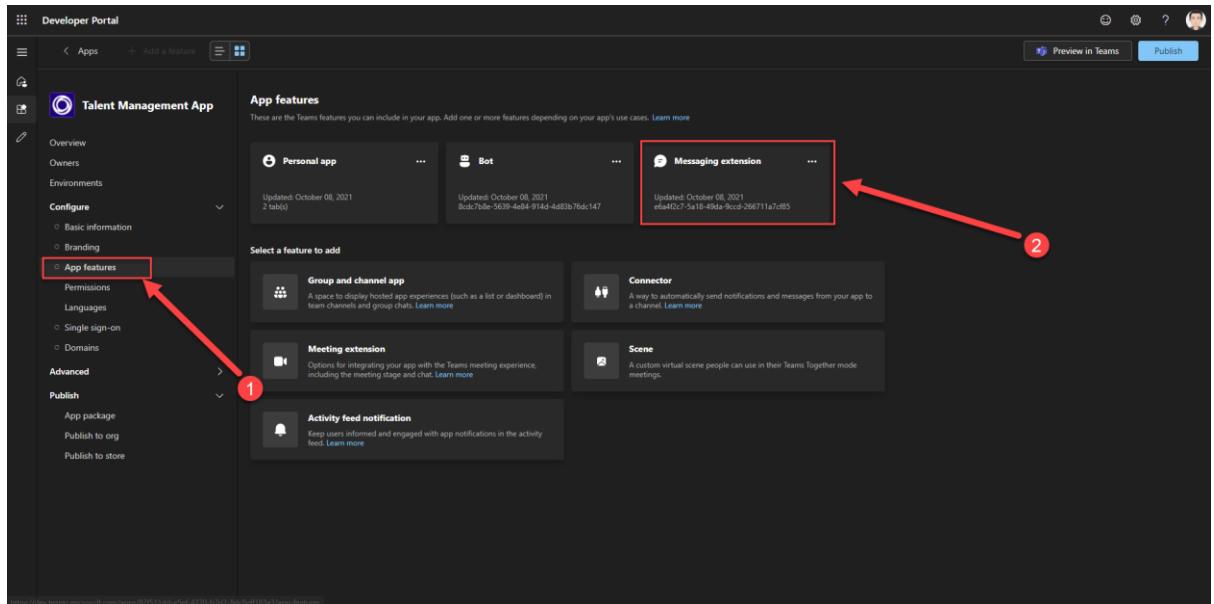
Open the TeamsTalentMgmtBot.cs file and set a breakpoint on line 40. Once you invoke the search by either using the compose message extension or the command box your breakpoint should be hit and you can inspect the values that make up the search command.

```
34     _botService = botService;
35     _invokeActivityHandler = invokeActivityHandler;
36 
37     public override async Task OnTurnAsync(ITurnContext turnContext, CancellationToken cancellationToken) => { await base.OnTurnAsync(turnContext, cancellationToken); }
38 
39     // Save any state changes etc.
40     protected override void SaveState(ICanvasControl turnContext)
41     {
42         _conversationsState.Save();
43         _userState.SaveChanges();
44     }
45 
46     protected override void OnTeamsSignin(ITurnContext turnContext, string user)
47     {
48         _invokeActivityHandler.Handle(signIn);
49     }
50 
51     protected override Task OnTeamsSignout(ITurnContext turnContext, string user)
52     {
53         _invokeActivityHandler.Handle(signOut);
54     }
55 
56     protected override Task OnTeamsMessage(ITurnContext turnContext, Activity activity)
57     {
58         _invokeActivityHandler.Handle(activity);
59     }
60 
61     protected override Task OnTeamsFileAttachment(ITurnContext turnContext, Attachment fileAttachment)
62     {
63         _invokeActivityHandler.Handle(fileAttachment);
64     }
65 
66     protected override Task OnTeamsFileAttachmentEventArgs(ITurnContext turnContext, FileAttachmentEventArgs args)
67     {
68         _invokeActivityHandler.Handle(args);
69     }
70 
71     protected override Task OnFileAttachmentEventArgs(AttachmentEventArgs args)
72     {
73         if (turnContext.Activity.Attachments != null)
74         {
75             return _botService.HandleFileAttachments(turnContext, cancellationToken);
76         }
77     }
78 
79     protected override Task OnFileAttachmentEventArgs(AttachmentEventArgs args, CancellationToken cancellationToken)
80     {
81         if (turnContext.Activity.Attachments != null)
82         {
83             return _botService.HandleFileAttachments(turnContext, cancellationToken);
84         }
85     }
86 
87     protected override Task OnFileAttachmentEventArgs(AttachmentEventArgs args, CancellationToken cancellationToken, bool cancel)
88     {
89         if (turnContext.Activity.Attachments != null)
90         {
91             return _botService.HandleFileAttachments(turnContext, cancellationToken);
92         }
93     }
94 
95     protected override Task OnFileAttachmentEventArgs(AttachmentEventArgs args, CancellationToken cancellationToken, bool cancel, bool cancelFile)
96     {
97         if (turnContext.Activity.Attachments != null)
98         {
99             return _botService.HandleFileAttachments(turnContext, cancellationToken);
100        }
101    }
102 }
```

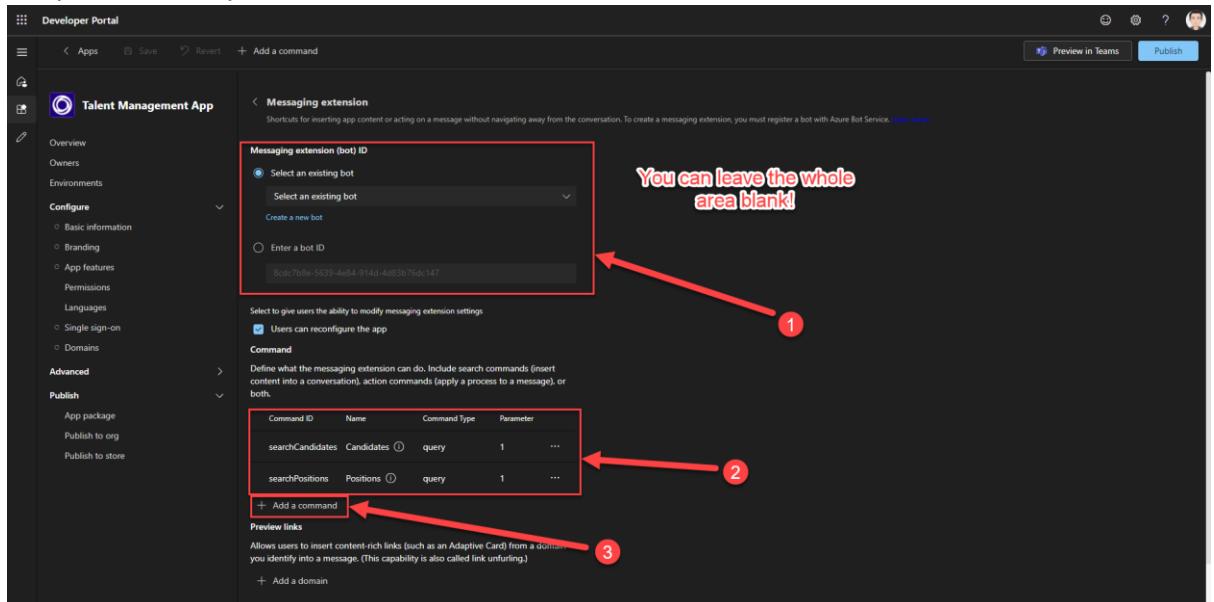
6) Implement a Message Extension Action

In this step we will add another message extension to perform and action instead of a search.

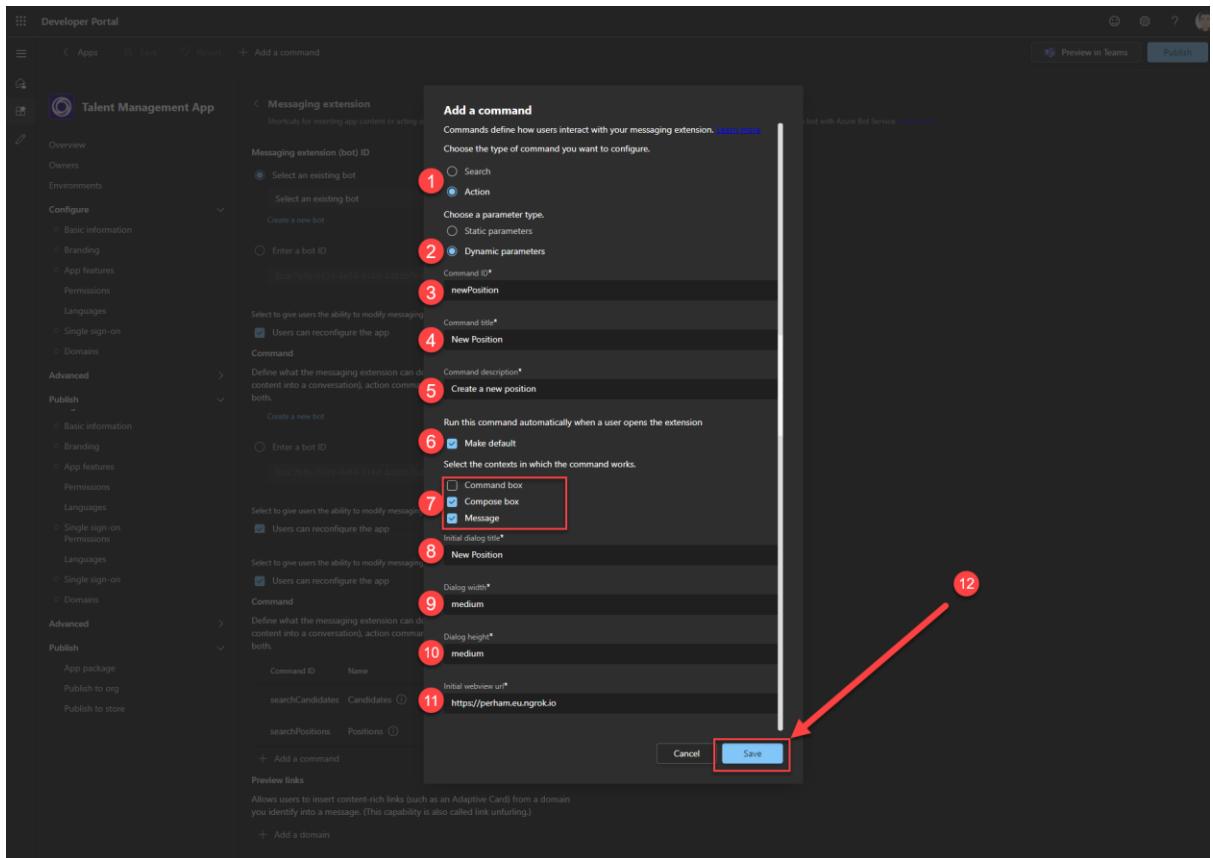
1. Open the Developer Portal, navigate to App Features and select Messaging extensions to edit the manifest.



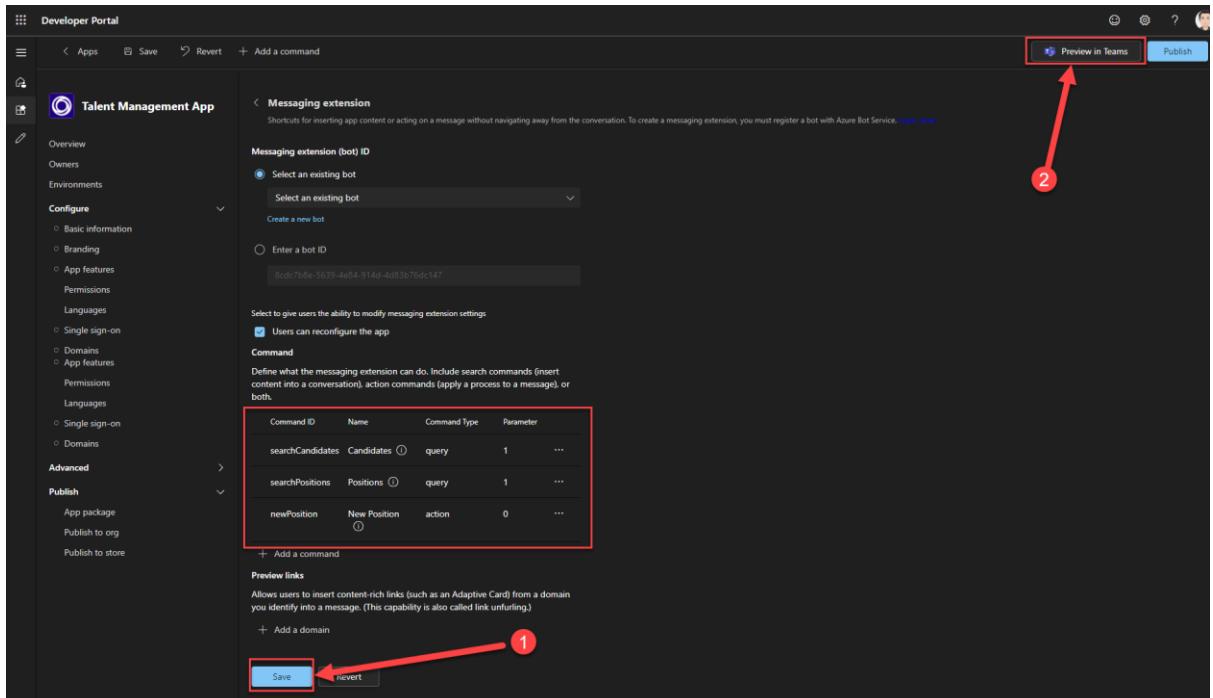
2. Here we will add a new command, if you can see the search commands listed then you are already in the context of your bot and don't need to select anything from the top section. If not you can enter your bot id as before. Then click **Add a command**.



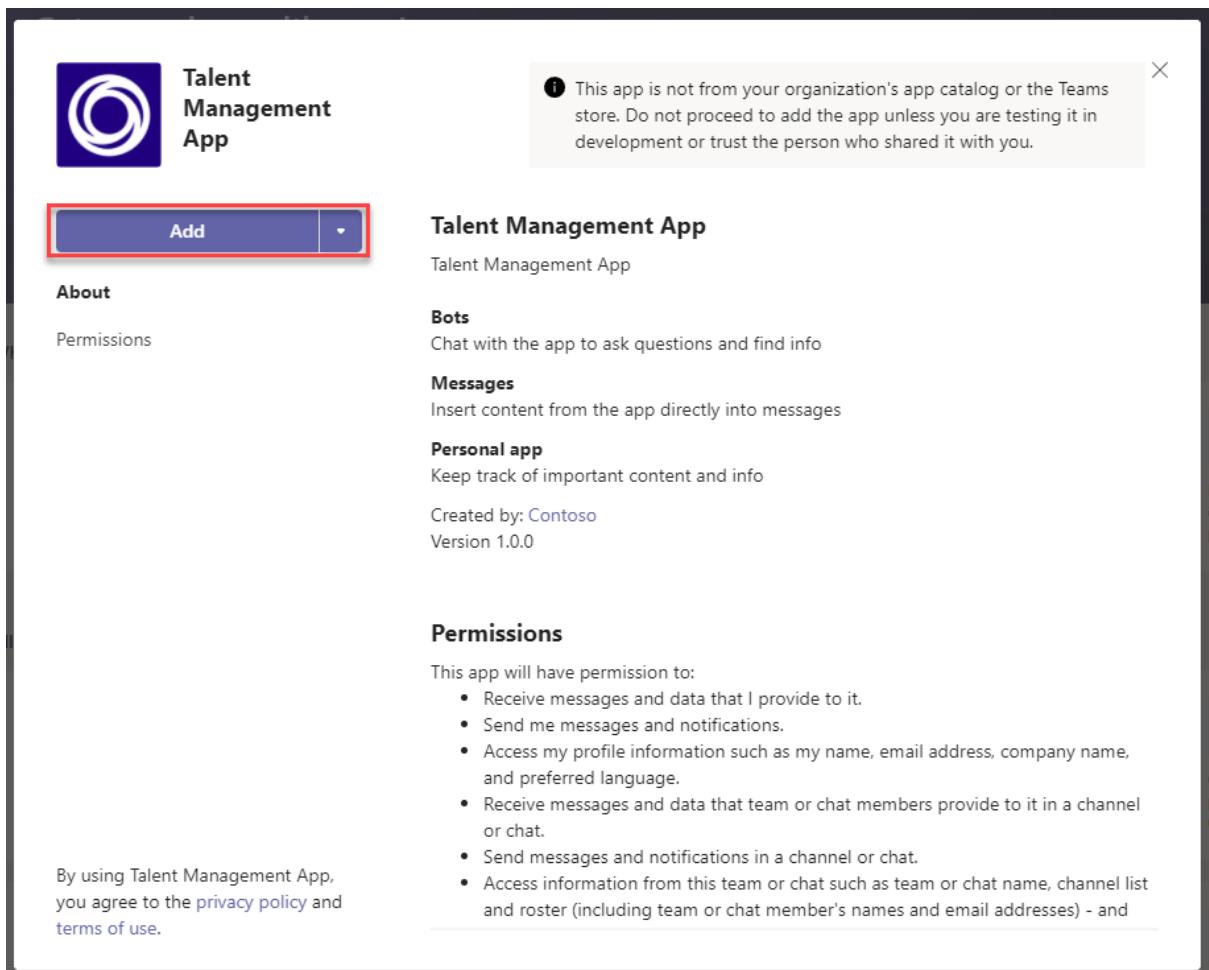
3. Complete the dialog as shown below, then click **Save**.



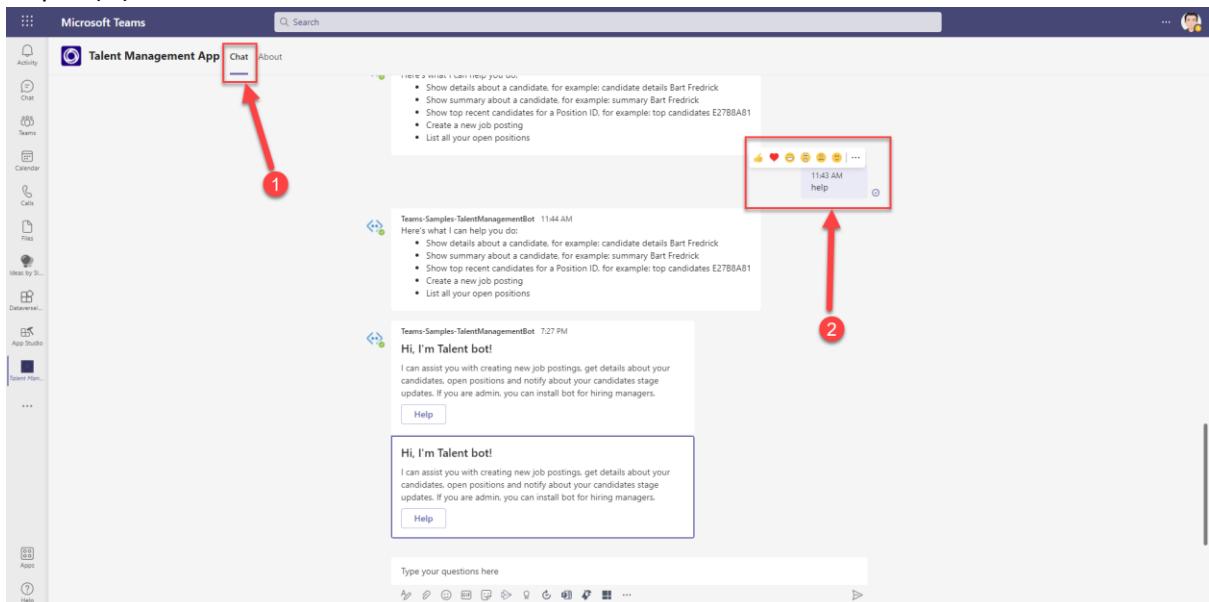
4. Ensure that you save the manifest by clicking **Save** and then we can test it. Click **Preview in Teams**.



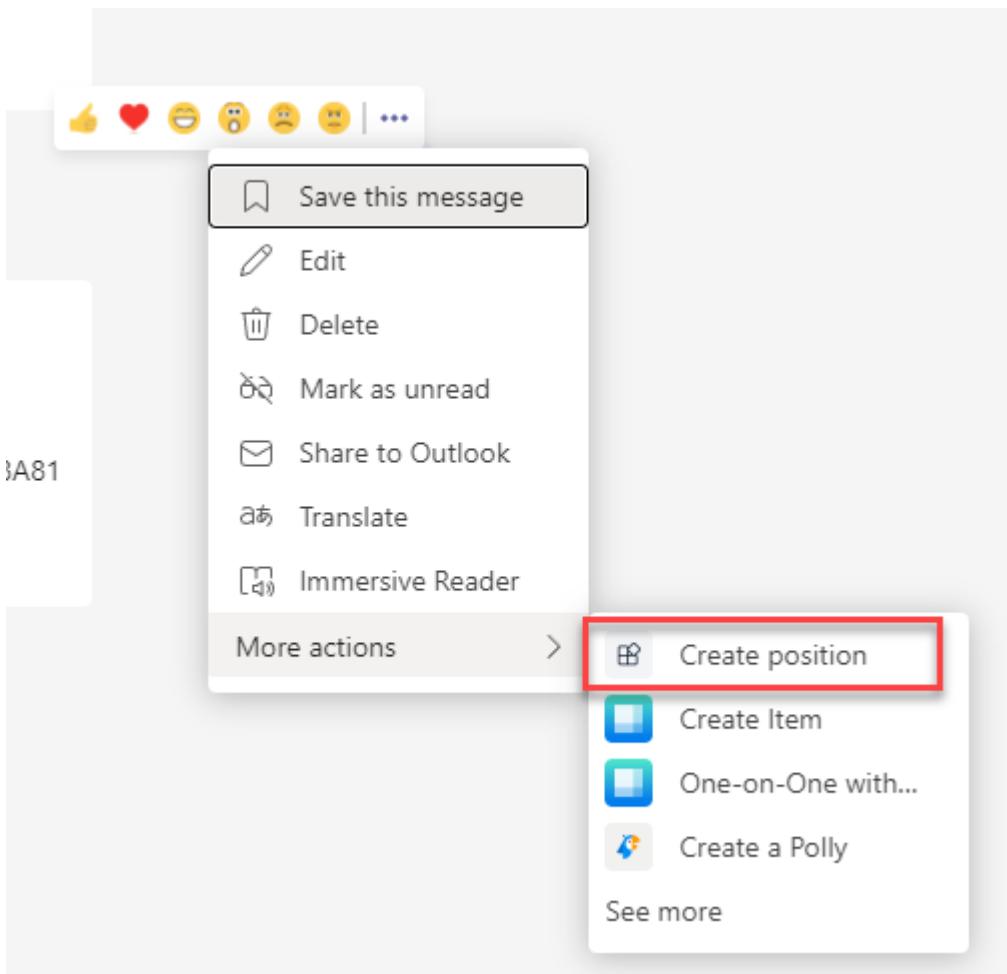
5. Again we will be prompted to add the app to Teams.



- Again ensure that you are in the Chat tab of your application. We configured the action to work on messages so we can invoke it by hovering over an existing message, and clicking the ellipsis (...).



- From the menu that's displayed, click **More actions** and then **Create position**.

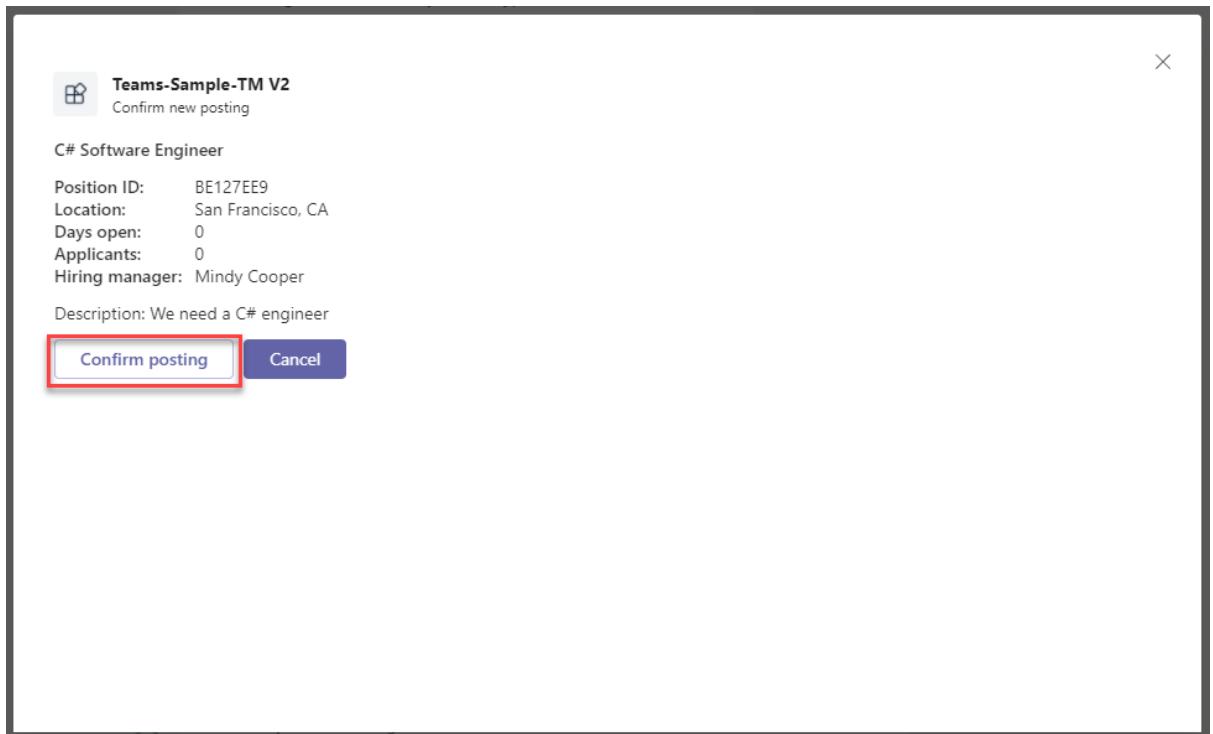


8. This will display a task module prompting for information pertaining to the position you are creating. Complete the form and click **Create posting**.

The screenshot shows a 'Create new job posting' task module. It has fields for Title (1), Level (2), Location (3), and Description (3). The 'Post by' field is set to Oct 8, 2021, and the 'Hiring manager' is Mindy Cooper. At the bottom is a 'Create posting' button (4), which is highlighted with a red box and an arrow pointing to it.

1 C# Software Engineer
2 10
3 San Francisco
3 We need a C# engineer
4 Create posting

9. You will then be asked to confirm the details before clicking **Confirm posting**.



- At this point the position has been saved and will be available through the tab or search functions shown earlier. It will also put an adaptive card in the compose box to allow you to send it to the chat if you wish!

The screenshot shows the Microsoft Teams interface with the 'Talent Management App' active in the chat. A message from the app displays a candidate's profile:

- Profile picture: Bart Fredrick
- Description: Ten years of experience in the software industry. Five years experience working at a software consulting firm.
- Current role: Senior Program Manager
- Location: New York, NY
- Steps: Applied
- Position applied: Senior Designer
- Date applied: Tuesday, September 11, 2018
- Phone number: +1 212 555 5445

Below this message, a bot message from 'Teams-Samples-TalentManagementBot' provides help:

- 11:40 AM
- Here's what I can help you do:
 - Show details about a candidate, for example: candidate details Bart Fredrick
 - Show summary about a candidate, for example: summary Bart Fredrick
 - Show top recent candidates for a Position ID, for example: top candidates E2788A81

An adaptive card is visible in the compose box, containing the following information:

Teams-Sample-TM V2

C# Software Engineer

Position ID: BE127EE9
Location: San Francisco, CA
Days open: 0
Applicants: 0
Hiring manager: Mindy Cooper

Description: We need a C# engineer

Congratulations! You have now successfully completed this lab. We hope that you found this lab, and the associated lab materials useful. We look forward to seeing what Teams Apps you build as a result of attending this lab!

Lab Complete.