Developing for Microsoft Teams

Lab Time: 60 minutes

Lab Folder: C:\Student\Modules\10_MicrosoftTeams\Lab

Lab Overview: In this lab, you will begin by configuring your Office 365 tenant to allow uploading custom apps into Microsoft Teams. After that, you will work through several lab exercises in which you will create and test a few custom apps for Microsoft Teams.

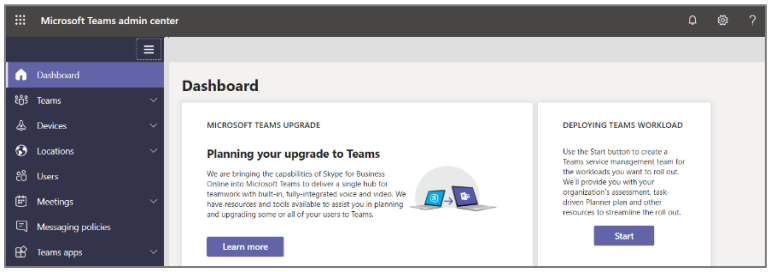
Exercise 1: Configure Your Environment for Microsoft Teams Development

In the first exercise, you will navigate to the Microsoft Teams admin and create a custom policy that allows for uploading apps.

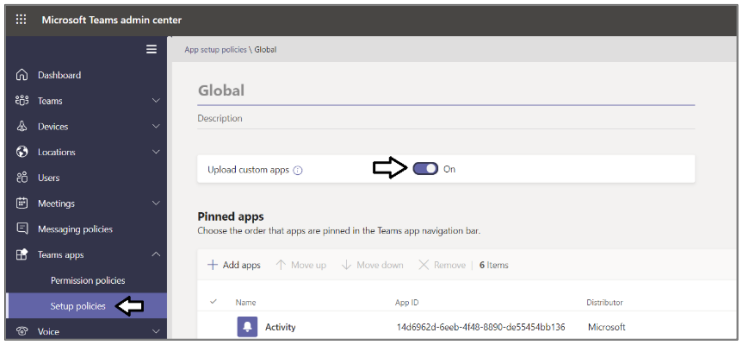
1. Navigate to the **Microsoft Teams admin center**.
   a) Using a browser, navigate to the following URL and login with your Office 365 user account.

   ```
   https://admin.teams.microsoft.com/dashboard
   ```

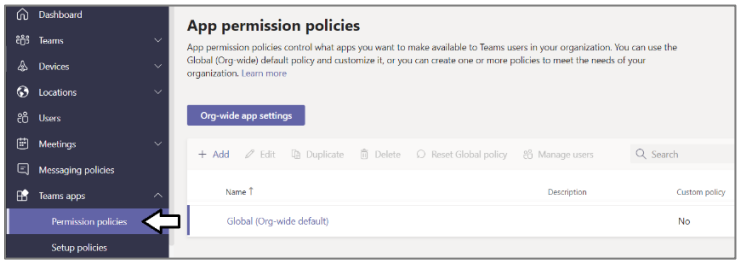
   b) You should now see the **Microsoft Teams admin center** as shown in the following screenshot.



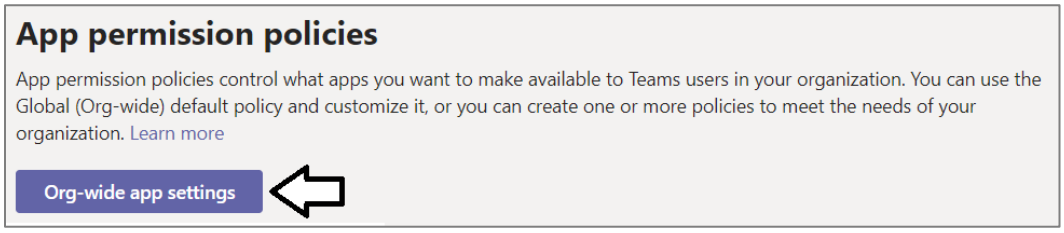
2. Enable the Setup policy to enable the uploading of custom apps.
   a) Navigate to **Teams apps > Setup policies** in the left navigation.
   b) Enable the **Upload custom apps** setting.



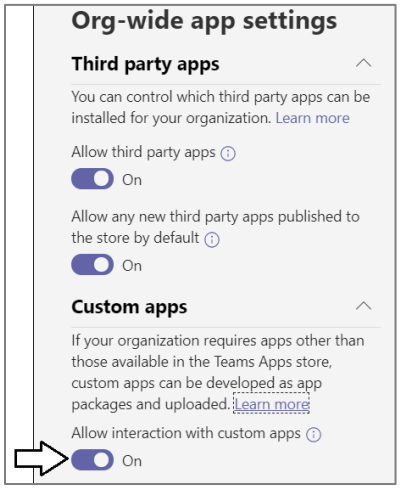
3. Inspect the **Global (Org-wide app default)** permission policy.
   a) Using the left navigation, navigate to **Teams apps > Permission policies**.
   b) In the **App permission policy** list, you should see there is one existing policy named **Global (Ord-wide default)**.



c) Click on the **Org-wide app settings** button to view to default org-wide policy



d) You should see that the **Custom apps** setting has been turned on by default.

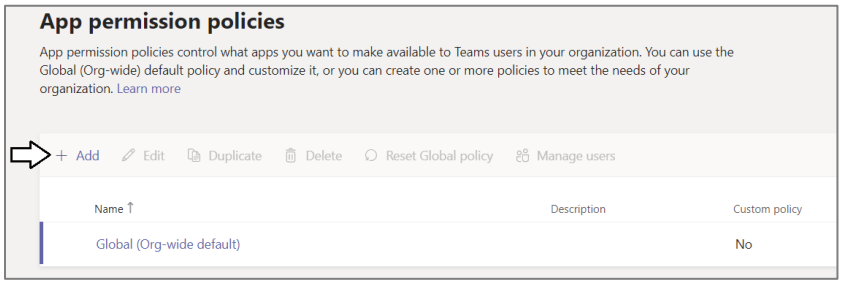


The **Custom apps** setting should be set to **On** by default. If the **Custom apps** setting is **Off**, turn it to **On**.

e) Click **Close** button to close the **Org-wide app settings** pan.

4. Add a new custom permissions policy to allow uploading custom apps in personal scope and d teams scope.

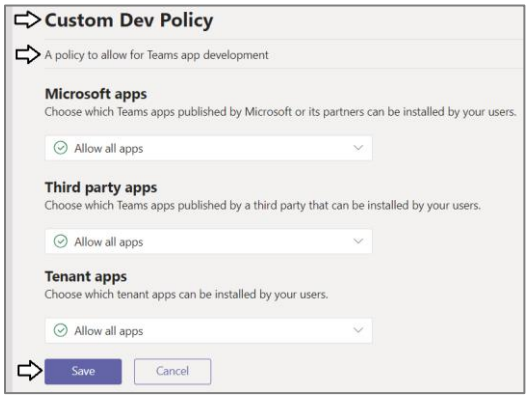
a) Click the **Add** button to display a form which allows you to create a new policy.



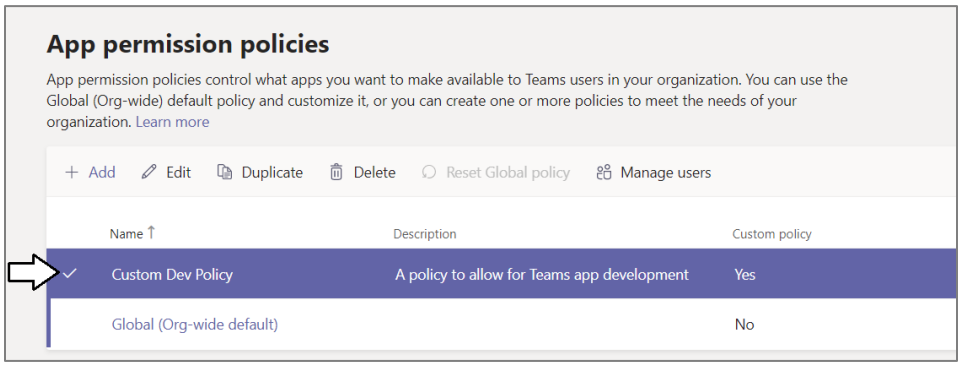
b) When the **App permission policy \ Add** form appears, place the cursor in the **Add app permission policy** textbox.

c) Type in a policy name of **Custom Dev policy** and a description of **A policy to allow for Teams app development**.

d) Leave all other setting with their default values and click **Save** to create the new policy.



e) When you are done, you should be able to see **Custom Dev Policy** in the **App permission policies** list.



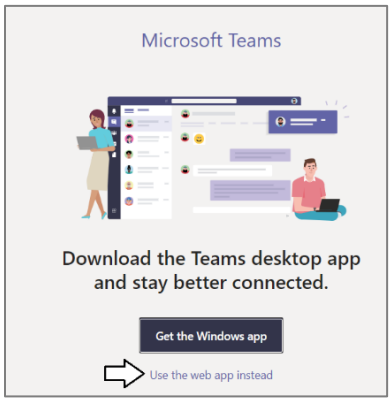
Now that you have added the custom app permission policy, you can upload app packages into Teams at personal scope or at teams scope. This helps to speed up development because it's easier and faster to upload and test apps during the development process.

5. Launch the Teams web app.

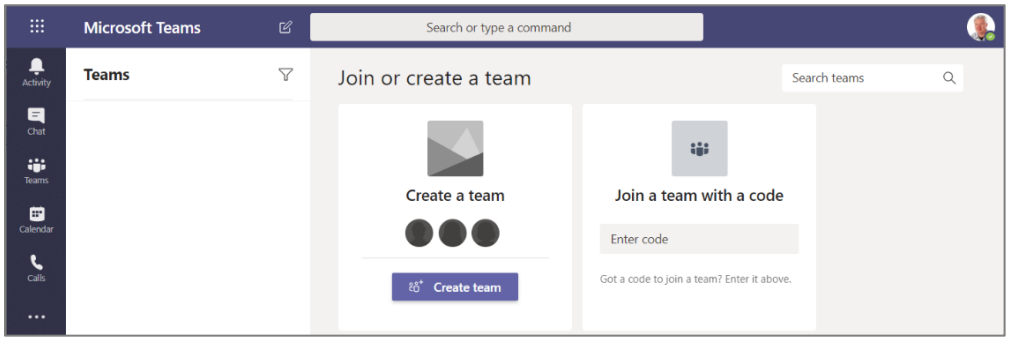
a) Navigate to the following URL to open the Teams web app.

```
https://teams.microsoft.com
```

b) You should now be prompted with a web page the encourages you to download the native Teams app for Windows. For now, bypass the option to **Get the Windows app** and click the link with the caption **Use the web app instead**.



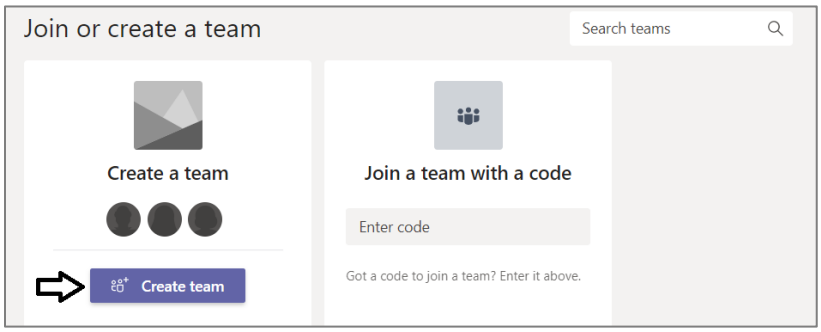
c) You should now see the **Microsoft Teams** web app.



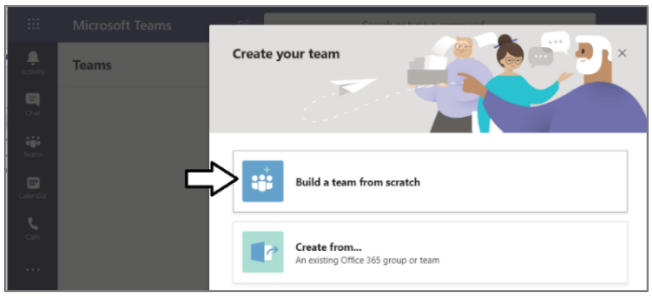
If this is the first time you have worked with Microsoft Teams in your development environment, there will not be any Teams that have been created yet. You must create a new team before you can develop and test Teams apps.

6. Create a new team.

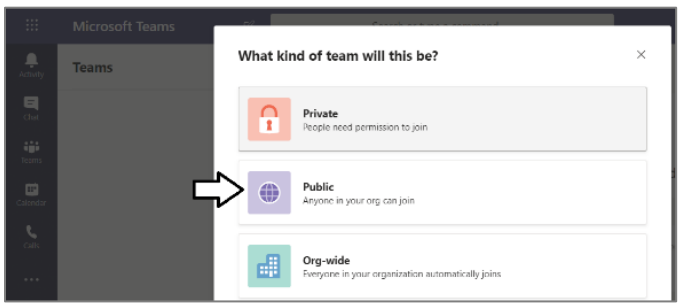
a) Click the **Create team** button.



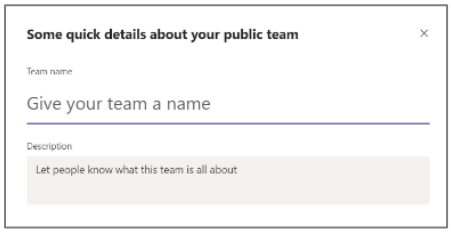
b) When prompted with the **Create your team** dialog, select **Build a team from scratch**.



c) When prompted with the **What kind of team will this be?** dialog, select **Public**.



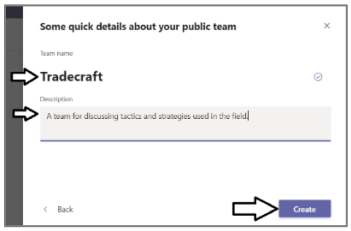
d) When prompted with the **Some quick details about your public team** dialog, place your cursor in the **Team name** textbox.



e) Add a **Team name** of **Tradecraft**.

f) Add a **Description** of **A team for discussing tactics an strategies in the field**.

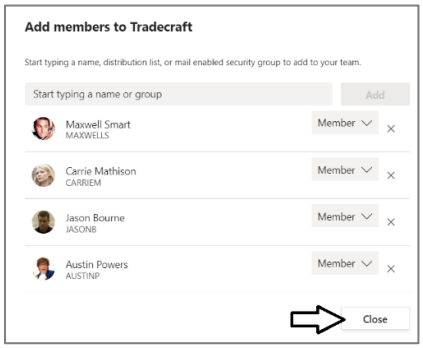
g) Click **Create** to create the new team.



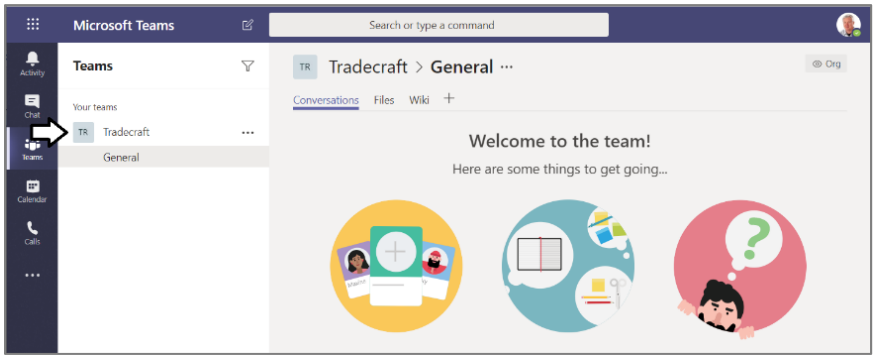
h) In the **Add members to Tradecraft** dialog, add a few users that you added to your tenant in lab 1.

i) Once you have added a few members, click the **Add** button.

j) Once you have added the members to your new team, click the **Close** button.



k) You should now see the **Tradecraft** team in the **Microsoft Teams** web app.



Note that a team is always created with a default channel named **General**. You can add additional channels to a team as needed.

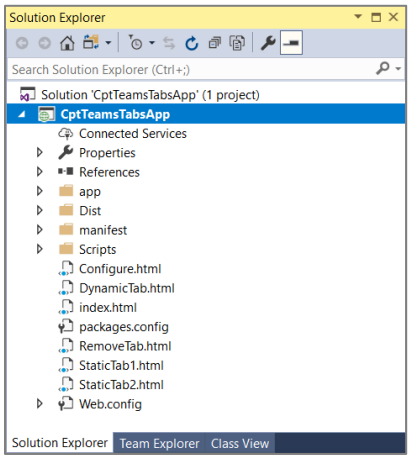
Exercise 2: Test a Microsoft Teams App with Custom Tabs

In this exercise, you will begin by working with a Visual Studio project with a simple custom Teams app with custom tabs that has already been created. The purpose of this exercise is to teach you how to test and debug a custom app in the Microsoft Teams environment.

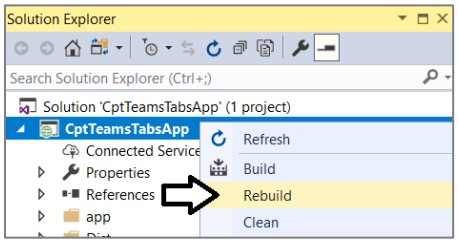
1. Open the **CptTeamsTabsApp** project in Visual Studio.
   a) Open the Visual **CptTeamsTabsApp** solution file at the following location using Visual Studio 2017 or Visual Studio 2019.

   ```
   C:\Student\Modules\10_MicrosoftTeams\Lab\CptTeamsTabsApp\CptTeamsTabsApp.sln
   ```

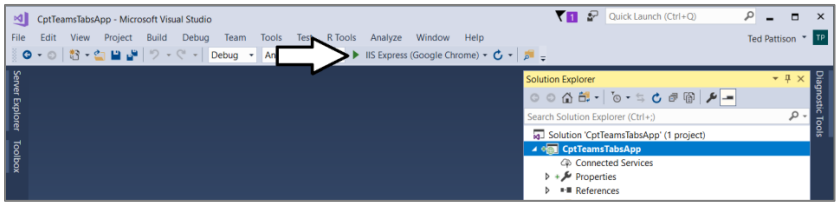
   b) Take a moment to review the structure of the **CptTeamsTabsApp** project.



   c) Right-click on the **CptTeamsTabsApp** project in solution explorer and click **Rebuild** to restore the project's NuGet packages



   d) Launch the project by pressing the **{F5}** key or pressing the **Start** button in the Visual Studio toolbar.



   e) A web page should appear at the URL of **http://localhost:3000**.

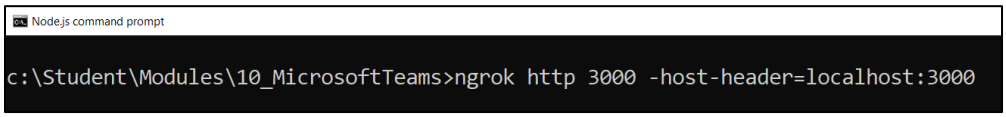
2. Download the **ngrok** utility (if you haven't already done so).
   a) Launch a browser and navigate to the following link.

   ```
   https://ngrok.com/download
   ```

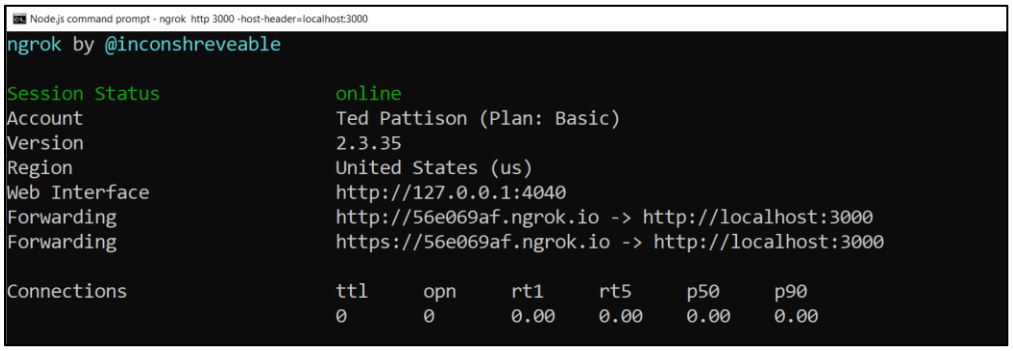
   b) Click the **Download for Windows** button to download a zip archive which contains **ngrok.exe**.
   c) Once you have downloaded the zip archive, extract **ngrok.exe** to a local folder that is in your SYSTEM path.
3. Use **ngrok** to start a session to make the **CptTeamsTabsApp** accessible from across the Internet.
   a) In a command prompt, run the **ngrok** utility to create a tunnel to the Visual Studio project running at **http://localhost:3000**.

   ```
   ngrok http 3000 -host-header=localhost:3000
   ```

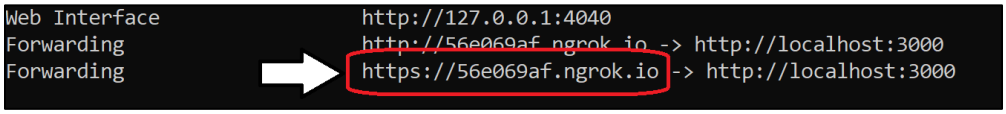
   b) The command you type and execute should match the following screenshot.



   c) Once you run the **ngrok** command, you should see output in the console matching the following screenshot.



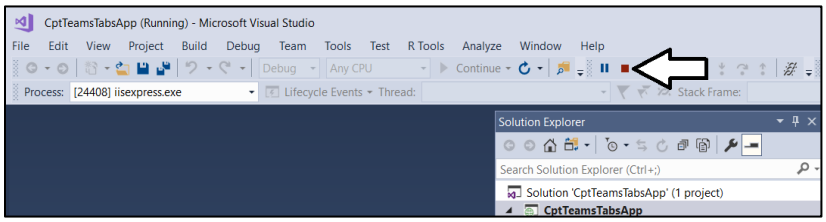
   d) Locate the forward URL that starts with **https**.



The URL displayed in this screenshot is **https://56e069af.ngrok.io**. Your will be different but still be in the format of **https://*.ngrok.io**.

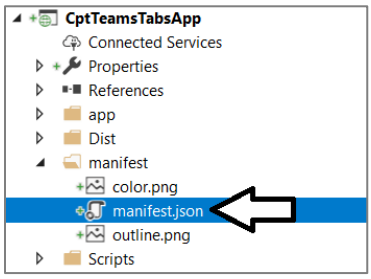
It is important to note that you are going to leave this **ngrok** session running throughout this entire lab exercise. If you stop this ngrok session and restart it, the URL will change.

   e) Return to the **CptTeamsTabsApp** project in Visual Studio and terminate the debugging session.



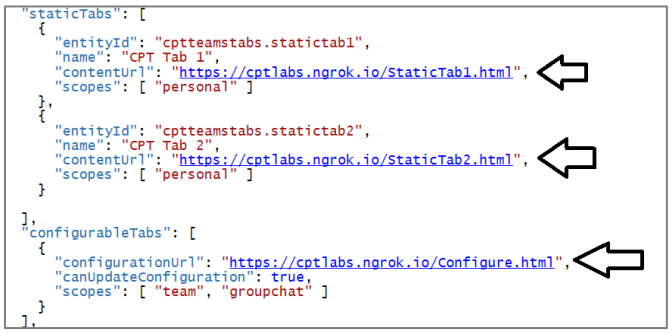
While it's important to leave the **ngrok** session running, you can stop and restart the Visual Studio debugging session for the **CptTeamsTabsApps** project without causing any problems. That's because Visual Studio will continue to use the same local URL of **http://localhost:3000**.

4. Update the Teams app manifest with the URL of your **ngrok** session.
   a) In the **CptTeamsTabsApp** project, locate the **manifest.json** file inside the **manifest** folder.
   b) Click on the **manifest.json** file to open it in an editor window.



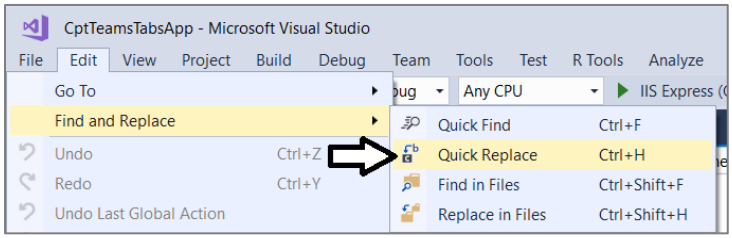
   c) You should see that the **manifest.json** contains the metadata for a teams app.

   d) You should see that the manifest defines two static tabs and one configurable (i.e. dynamic) tab

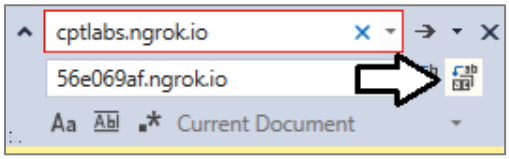


Note that the manifest currently defines URLs with a host of **cptlabs.ngrok.io**. You will now run a search and replace operation on **manifest.json** to replace the host **cptlabs.ngrok.io** with the host of your currently running **ngrok** session.

   e) With **manifest.json** selected as the active window, select **Find and Replace > Quick Replace** from the **File** menu.



f) Run the **Replace All** command to replace **cptlabs.ngrok.io** host name with the host name of your running **ngrok** session.

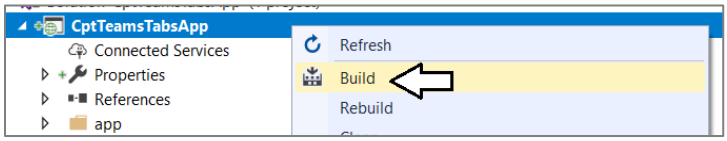


g) You should be able see four places in **manifest.json** where the host name has been replaced

h) Save your changes and close **manfiest.json**.

5. Build the app manifest for the the **CptTeamsTabsApp** project.

a) Right click on the **CptTeamsTabsApp** project and select the **Build** command.

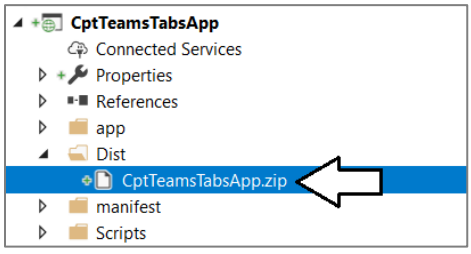


Note that the Visual Studio project file named **CptTeamsTabsApp.csproj** has been extended with post-build action to build the app package named **CptTeamsTabsApp.zip** by compressing all files found in the project's **manifest** folder. Here is a glimpse of what the post-build action looks like inside **CptTeamsTabsApp.csproj**.

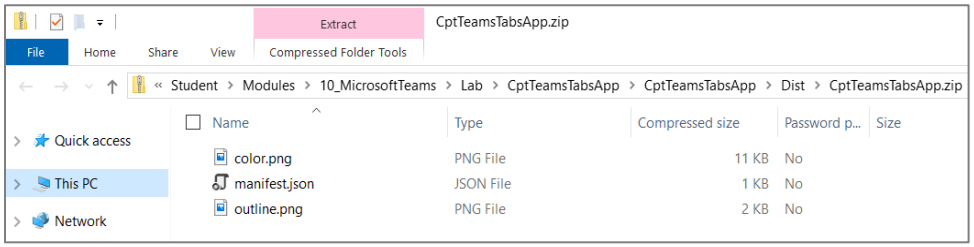
```
<PostBuildEvent>
  powershell.exe Compress-Archive -Path \"$(ProjectDir)Manifest\*\"
                                  -DestinationPath \"$(ProjectDir)Dist\CptTeamsTabsApp.zip\"
                                  -Force
</PostBuildEvent>
```

b) Once you have run the **Build** command without errors, look inside the **Dist** folder and locate **CptTeamsTabsApp.zip**.

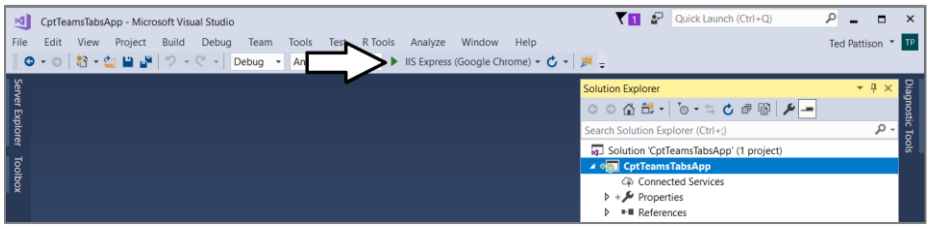
c) Double click on the zip archive named **CptTeamsTabsApp.zip** to open it in Windows explorer.



d) You should see the zip archive for the app package contains **manifest.json** long with two PNG files with icon image.

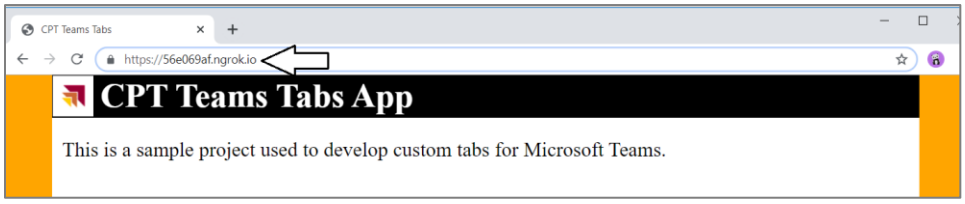


e) Launch th **CptTeamsLabsApp** project by pressing the **{F5}** key or pressing the **Start** button in the Visual Studio toolbar.



f) The project should launch in the browser and display local URL of **https://localhost:3000**.

g) Place your cursor in the browser address bar and replace the URL with the URL to your grok session and press **Enter**.



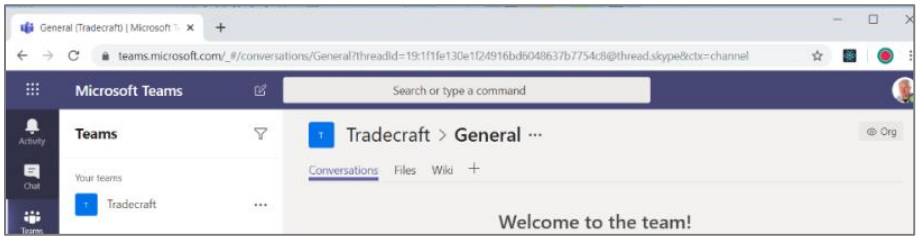
You should be able to verify that the **CptTeamsTabsApp** web site is accessible through the **ngrok** session URL in addition to being accessible through https://localhost:3000.

6. Return the the Microsoft Teams web app,

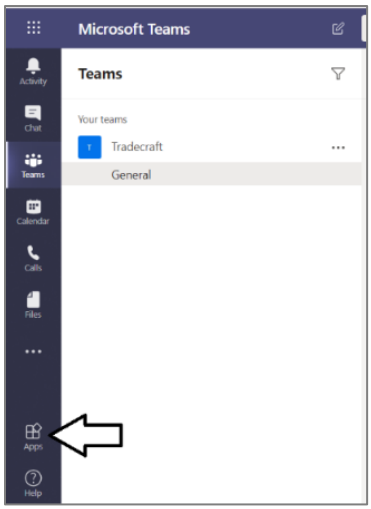
a) Navigate to the following URL to open the Teams web app.

```
https://teams.microsoft.com
```

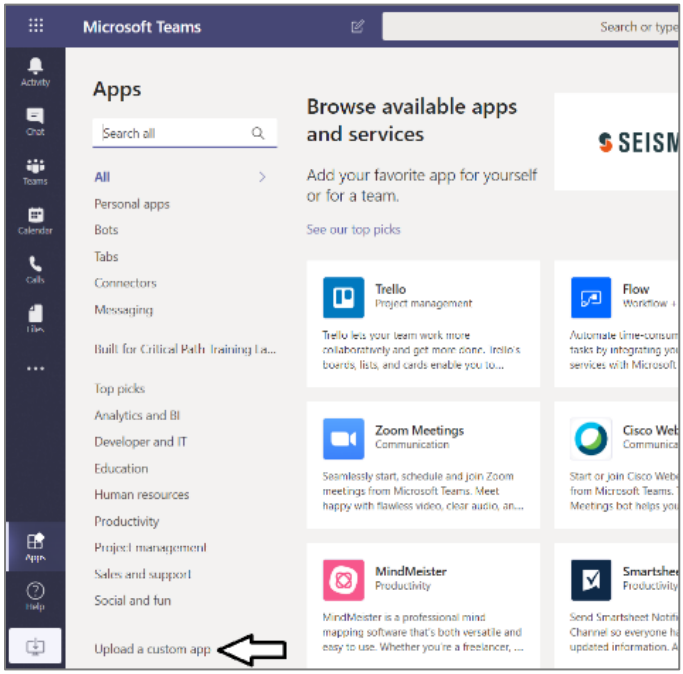
b) You should now see the Microsoft Teams web app.



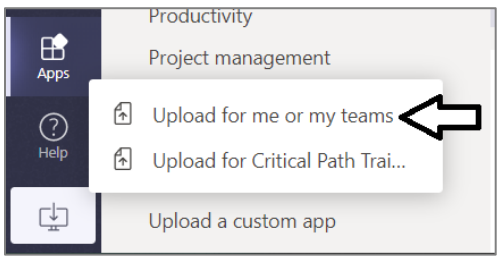
7. Install **CptTeamsTabsApp** to create a static page at personal scope.
   a) In the Microsoft Teams web app, Click on the Apps button in the left navigation.



   b) Click the **Upload a custom app** link at the bottom of the Browse available apps and services page.

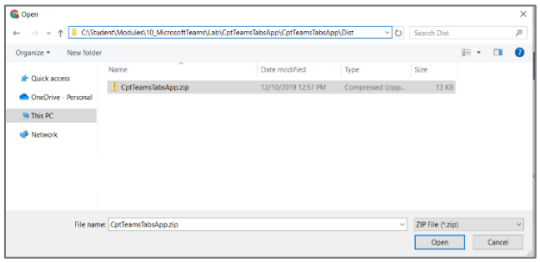


   c) Select the **Upload for me or my teams** menu command.

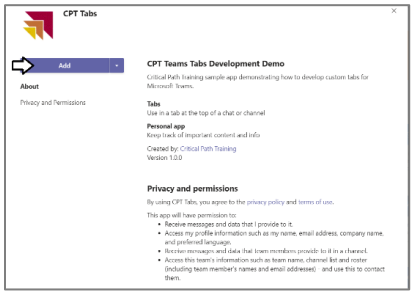


d) Select the app package at the following location.

```
C:\Student\Modules\10_MicrosoftTeams\Lab\CptTeamsTabsApp\CptTeamsTabsApp\Dist\CptTeamsTabsApp.zip
```

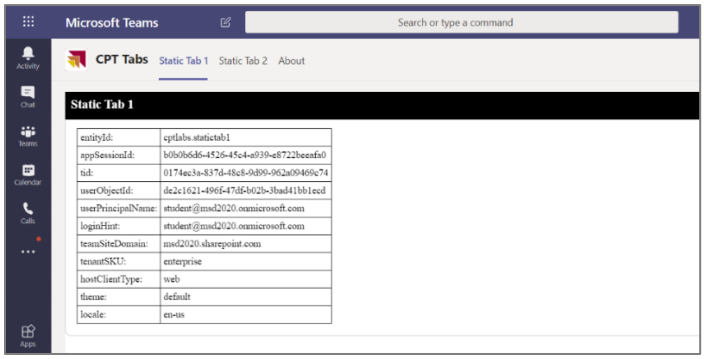


e) You should now see a dialog with information from the app manifest. Click **Add** to install the app as a personal tab.

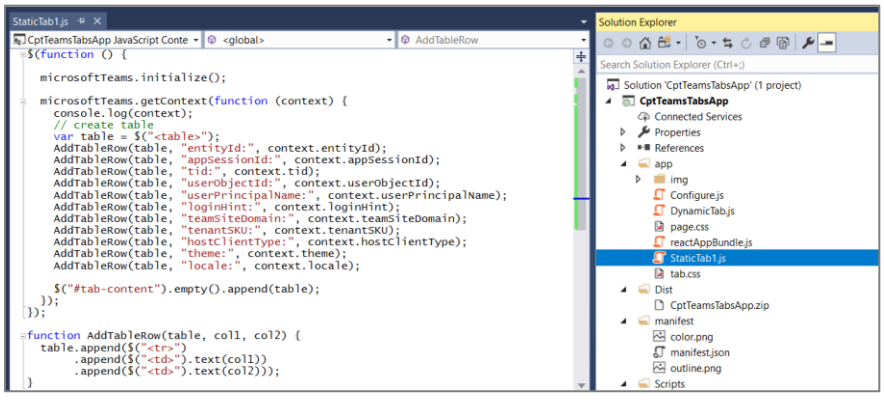


f) You should now see tabs for two static pages at personal scope.

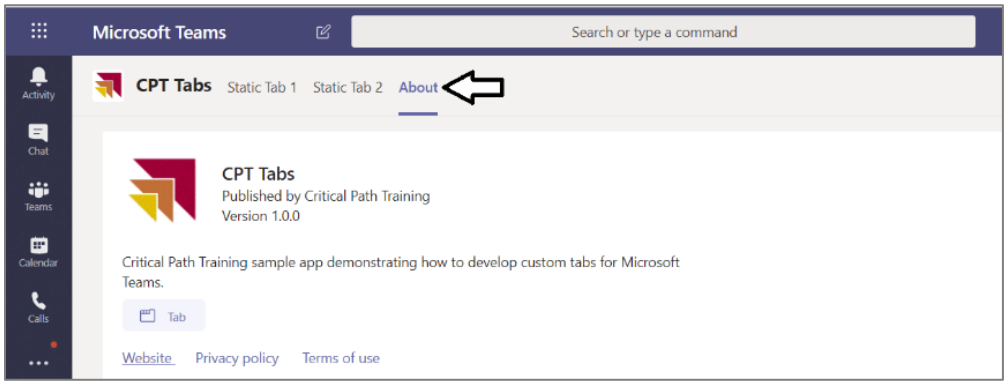
g) Examine the first static tab with the tab name of **Static Tab 1**.



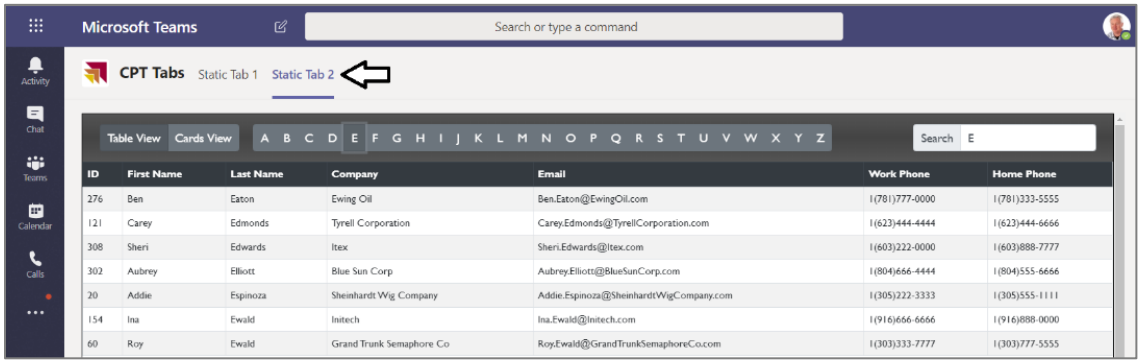
h) You can see the Teams JavaScript SDK code that populates **Static Tab 1** in the project file named **StaticTab1.js**.



i) Inspect the About to see the information it display from the app manifest.

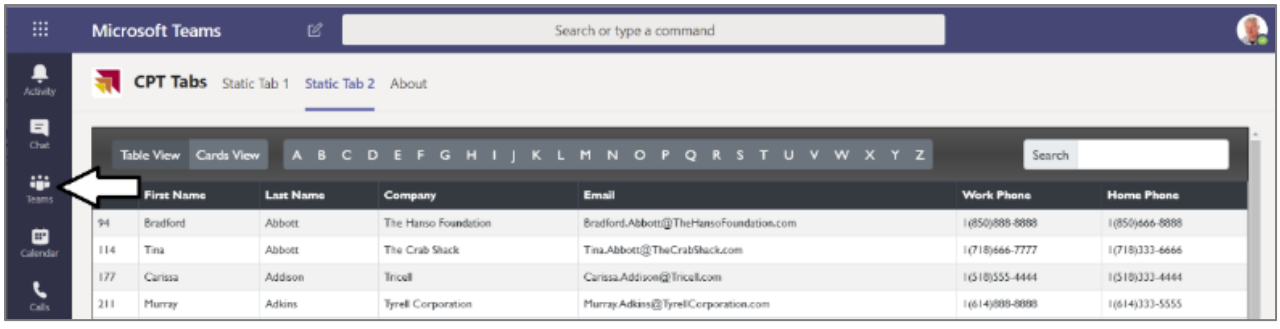


j) Inspect Static Tab 2 which uses a react.js to provide a user experience.



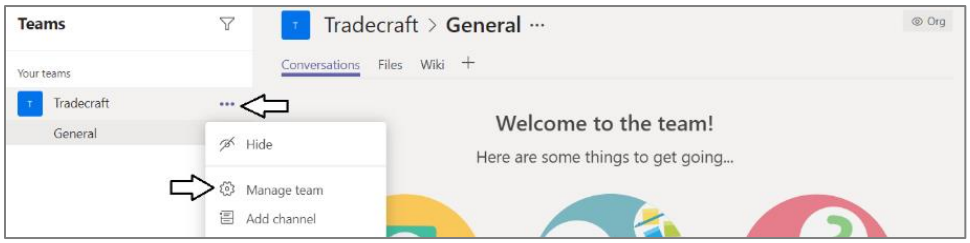
k) Switch **Static Tab 2** over to **Cards View**.

l) After you have tested the personal tabs, return to the Teams view by clicking the **Teams** link in the left navigation.

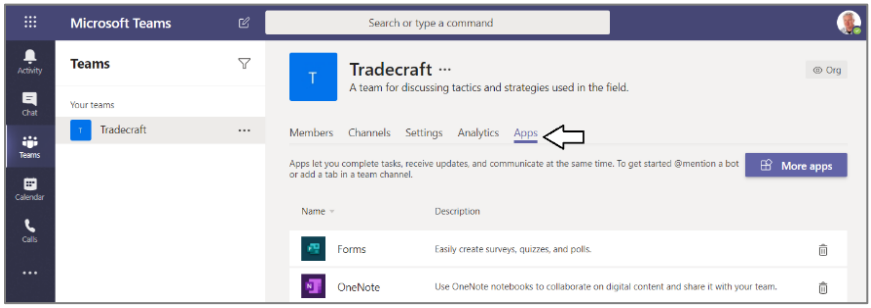


8. Install a configurable tab at teams scope.

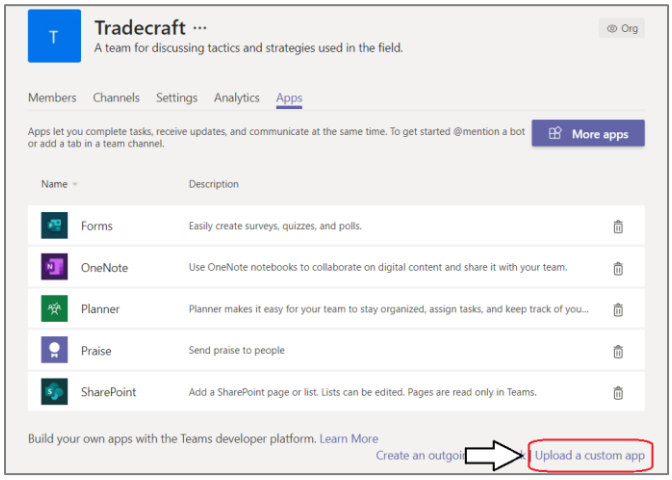
a) Use the dropdown ellipse menu of the **Tradecraft** team to select the **Manage Teams** command.



b) Click the **Apps** tab of the **Tradecraft** team.

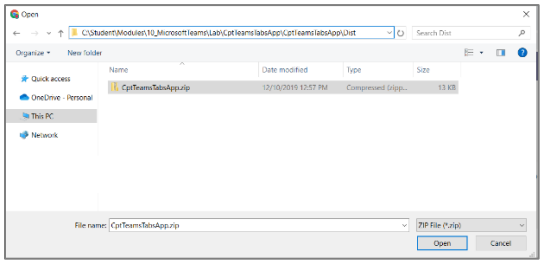


c) Locate and click the **Upload a custom app** link in the bottom left corner.

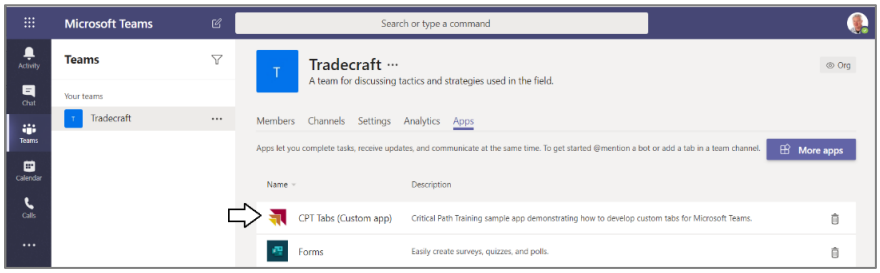


If you don't see the **Upload a custom app** link, you need to adjust the policy setting in the Microsoft Team admin center.

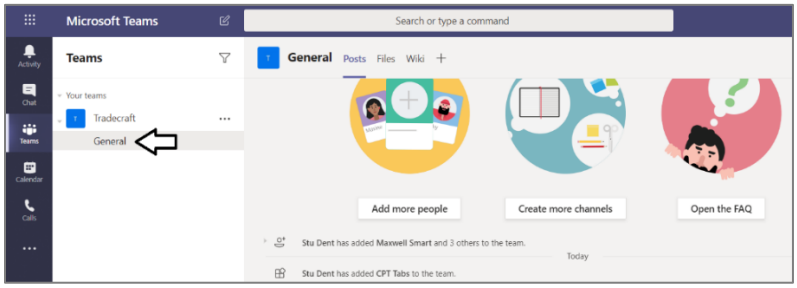
d) Upload the same app package you uploaded in a previous step named **CptTeamsTabsApp.zip**.



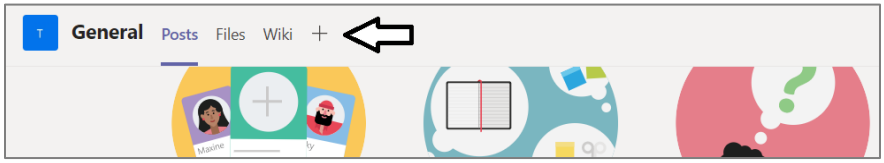
e) You should see that the app has now been installed within the scope of the **Tradecraft** team.



f) Navigate to the **General** channel of the **Tradecraft** team.

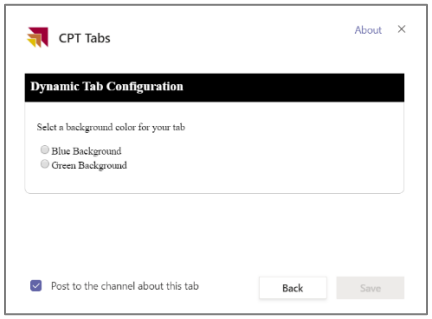


g) Click the **+** button to ad a new tab.

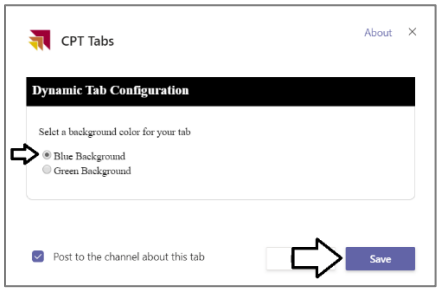


h) Select the **CPT Tabs** app to provide the new tab dynamic tab.

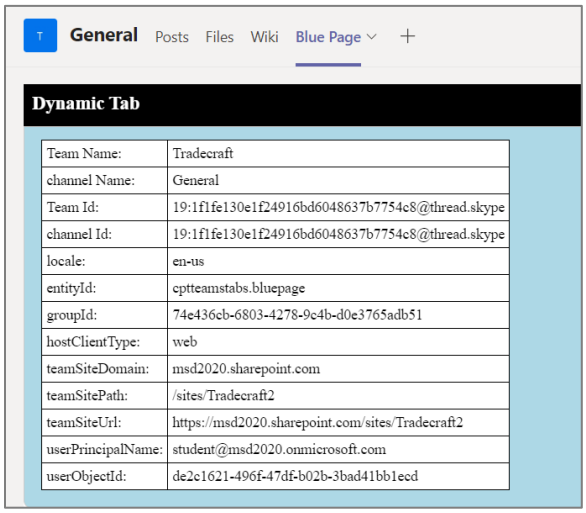
i) You should now see the configuration page used to create a dynamic tab.



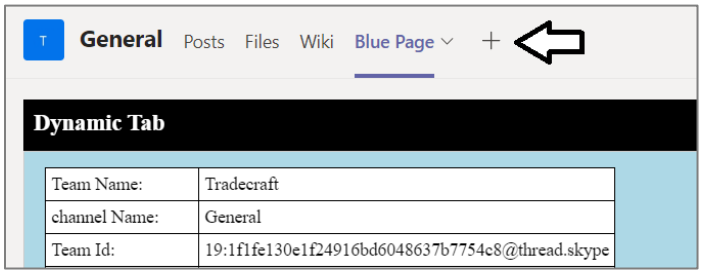
j) Select **Blue Background** and then click **Save**.



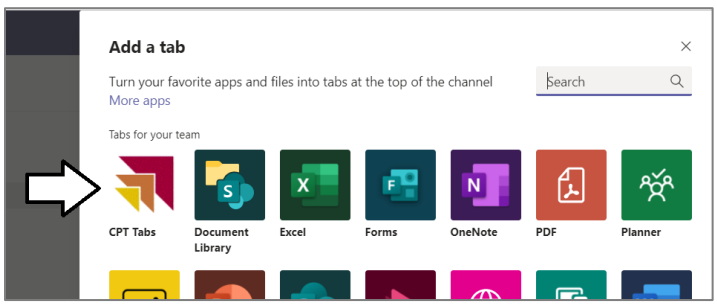
k) You should now see a dynamic page with a table of contextual information about the tab.



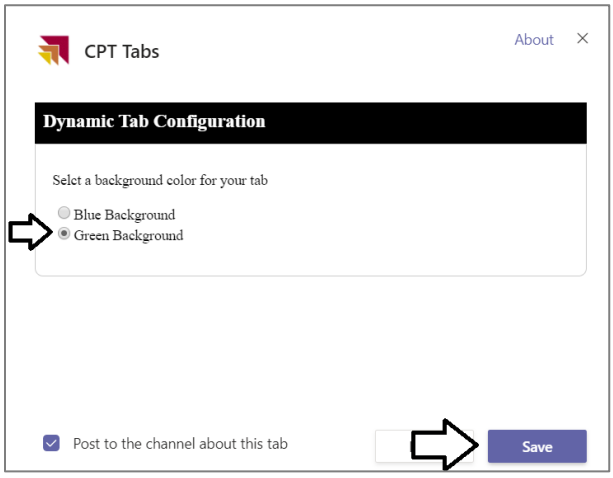
l) Click the **+** button to add a second page from the same configurable tab.



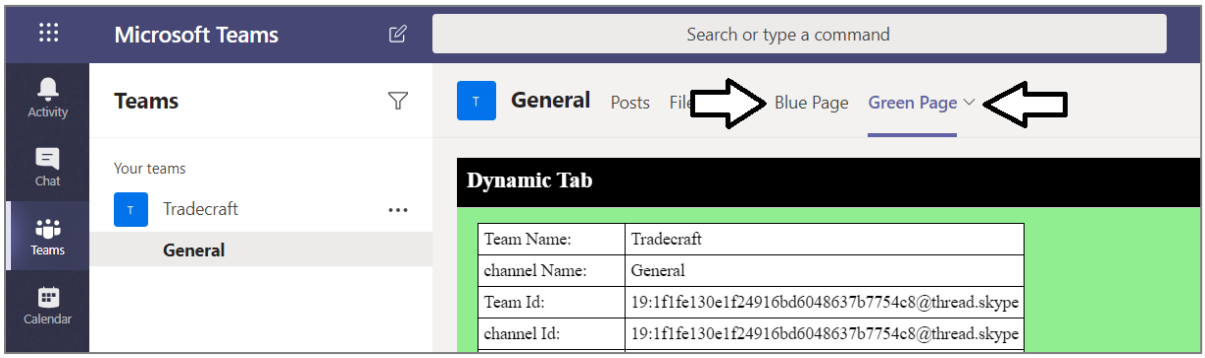
m) Select the **CPT Tabs** app to provide the new tab dynamic tab.



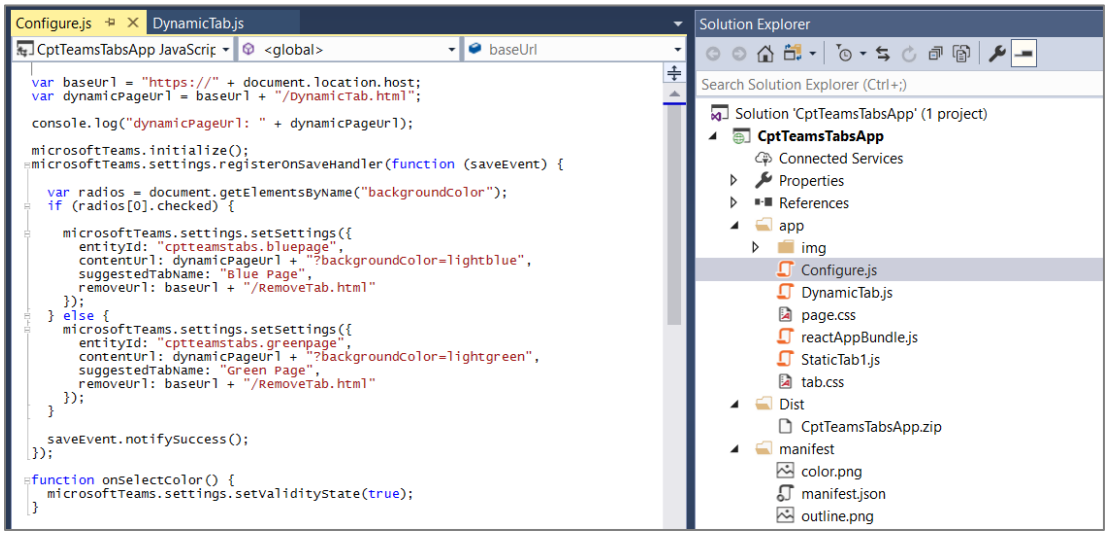
n) This time select **Green Background** and click **Save**.



o) You should now have two dynamics tabs with a blue page and a green page.



p) Inspect the project files named **Configure.js** and **DynamicTab.js** to see how these dynamic tabs are implemented.



9. You are now done with this exercise.
   a) Return to Visual Studio and terminate the debugging session.
   b) Return to the console and terminate the ngrok session.

Exercise 3: Create a Custom Teams App using Node.js and Visual Studio Code

This is a lab authored by the Microsoft product team.

1. Install the Node.JS packages required for working with SharePoint Framework.
   a) Launch the Node.JS command prompt.
   b) Run the following **npm** command to globally install the packages for **gulp** version 3 and the Yeoman Generator (**yo**).

```
npm install -g yo gulp-cli typescript
```

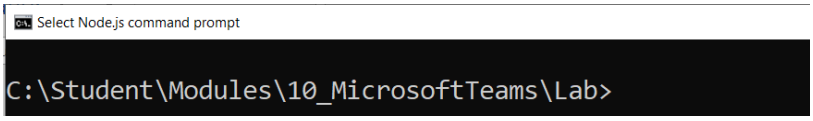
   c) Execute the following **npm** command to globally install the yo template for creating SharePoint Framework projects.

```
npm install -g generator-teams
```

2. Create a new SPFx project named **spfx-lab**.
   a) From the Node.JS command prompt, run the following command to set your current folder to the folder for this lab.

```
cd C:\Student\Modules\10_MicrosoftTeams\Lab
```

   b) The current directory for the console should now be at the folder for this lab inside the **Student** folder.



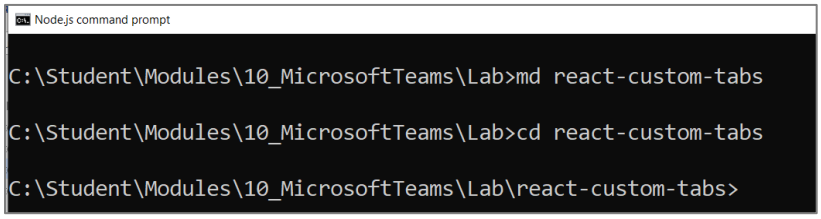
   c) Type the following command and execute it by pressing **Enter** to create a new folder for your project.

```
md react-custom-tabs
```

   d) Type the following command and execute it by pressing **Enter** to move to the current directory into the new folder.

```
cd react-custom-tabs
```

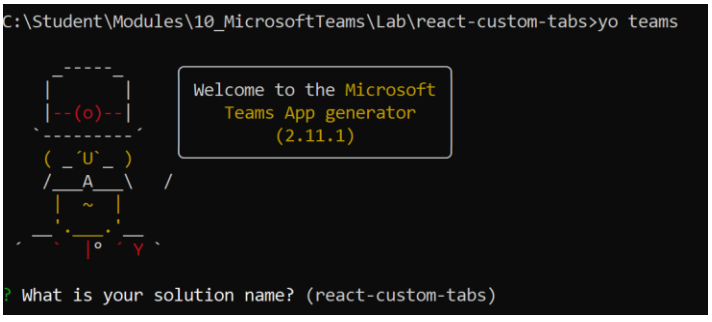
   e) The current directory for the console should now be located at the new folder you just created named **react-custom-tabs**.



   f) Type the following command and execute it to launch the Yeoman generator with the SharePoint Framework project template.

```
yo teams
```

   g) When prompted with **What is your solution name?**, press **Enter** to accept the default value which is the name of the folder.



h) When prompted, accept the default value of **react-custom-tabs** as your solution name and press **Enter**.

i) Select **Use the current folder for the file location** and select **Enter**.

j) Accept the default value of **react custom tabs** as the solution name and press **Enter**.

```
? What is your solution name? react-custom-tabs
? Where do you want to place the files? Use the current folder
? Title of your Microsoft Teams App project? (react custom tabs)
```

k) Select Use the current folder for Where do you want to place the files?.

l) Enter teams app1 as the Title of your Microsoft Teams App project.

m) Enter your name and press Enter.

n) Select v1.5 as the manifest version you would like to use and press Enter.

o) Enter a Microsoft Partner Id if appropriate

```
? Title of your Microsoft Teams App project? react custom tabs
? Your (company) name? (max 32 characters) Acme Corp
? Which manifest version would you like to use? v1.5
? Enter your Microsoft Partner Id, if you have one? (Leave blank to skip)
```

p) Accept the default selection of Tab for what you want to add to your project and press Enter.

```
? What features do you want to add to your project? (Press <space> to select, <a> to toggle a
>(*) A Tab
 ( ) A Bot
 ( ) An Outgoing Webhook
 ( ) A Connector
 ( ) A Message Extension Command
 ( ) Localization support
```

q) Enter https://tbd.ngrok.io as the URL where you will host this tab and press Enter. You will change this URL later in the exercise.

r) Enter n and press Enter when prompted to include a Test framework and initial tests.

s) Enter n and press Enter when prompted to use Azure Application Insights to telemetry.

t) When prompted for the **Default Tab name**, enter a value of **Custom React Tab** and press Enter.

```
? The URL where you will host this solution? https://abcd1234.ngrok.io
? Would you like to include Test framework and initial tests? No
? Would you like to use Azure Applications Insights for telemetry? No
? Default Tab name? (max 16 characters) Custom React Tab
```

u) When prompted for Tab Type, select **Configurable** and press Enter.

v) When prompted for the Scope of the Tab, select In a Team and press Enter.

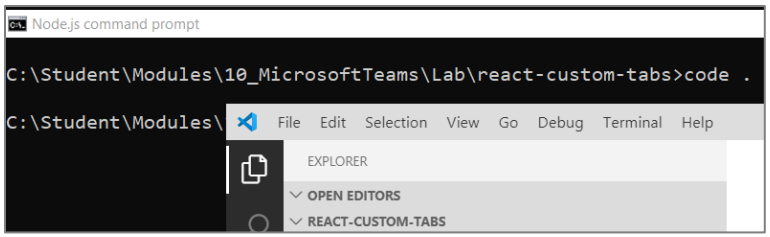
w) When prompted Tab to be available in SharePoint Online, enter n and press Enter.

x) The yo generator will run and produce this.

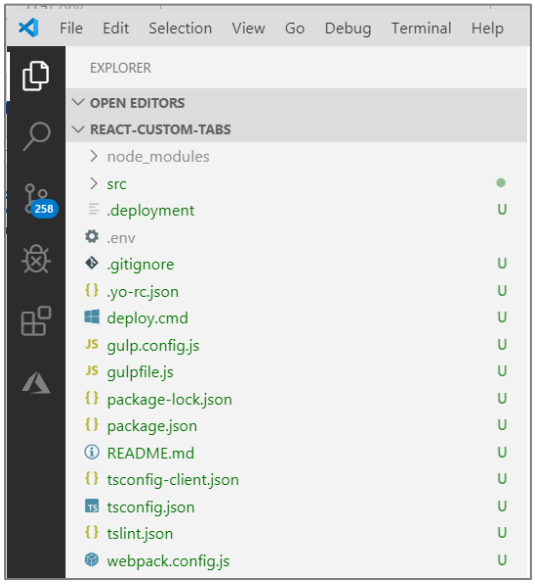
```
added 1181 packages from 874 contributors and audited 22296 packages in 51.123s
found 8 vulnerabilities (2 low, 1 moderate, 5 high)
  run `npm audit fix` to fix them, or `npm audit` for details
Thanks for using the generator!
Have fun and make great Microsoft Teams Apps...

C:\Student\Modules\10_MicrosoftTeams\Lab\react-custom-tabs>
```

3. Open the project in Visual Studio Code.

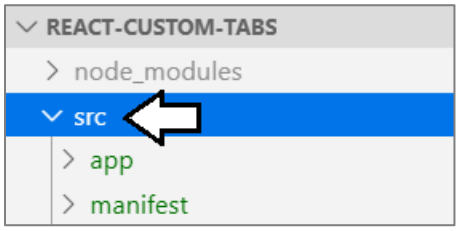
   a) Type and execute the command **code .** to open the project in Visual Studio Code.



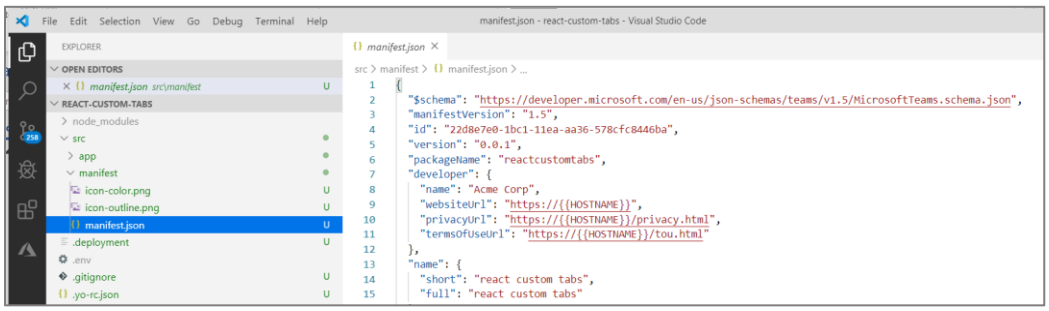
   b) Take a moment to review the structure of the new project.



   c) Expand the **src** folder.



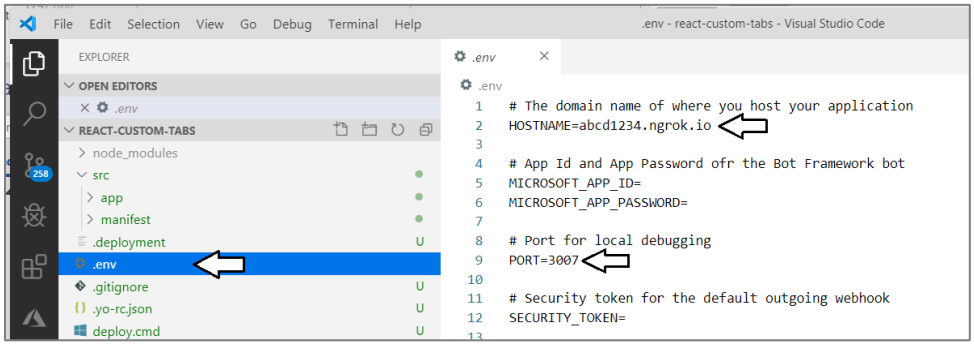
   d) Inspect the **manifest.json** file which contains the app manifest.



e) See what has already been defined in the **configurableTabs** section.

```
"configurableTabs": [
  {
    "configurationUrl": "https://{{HOSTNAME}}/customReactTab/config.html",
    "canUpdateConfiguration": true,
    "scopes": [
      "team"
    ]
  }
],
```

f) Open the **.env** file to see what's inside. There is no need to modify this file.



4. Start up the application by running the **gulp ngrok-serve** command.

a) Open the Visual Studio Code Terminal and execute the following command.

```
npx gulp ngrok-serve
```

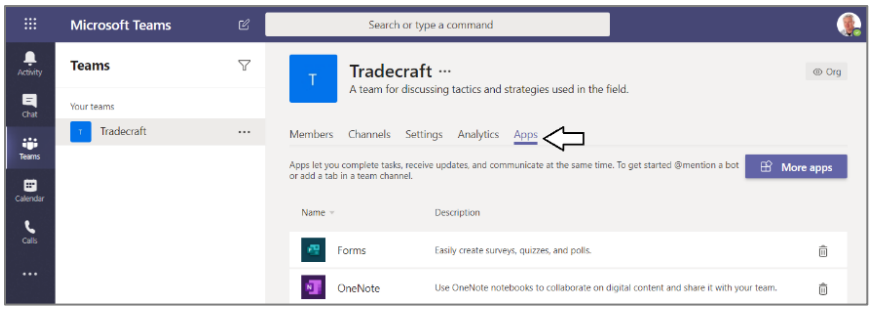
b) The command will start up an ngrok session and then rebuild the app manifest with the URL for the new ngrok session.

5. Upload the package for the new custom app.

a) Return to the Microsoft Teams web app and navigate to the Teams view to see the **Tradecraft** team.

b) Use the dropdown ellipse menu of the **Tradecraft** team to select the **Manage Teams** command.

c) Click the **Apps** tab of the **Tradecraft** team.

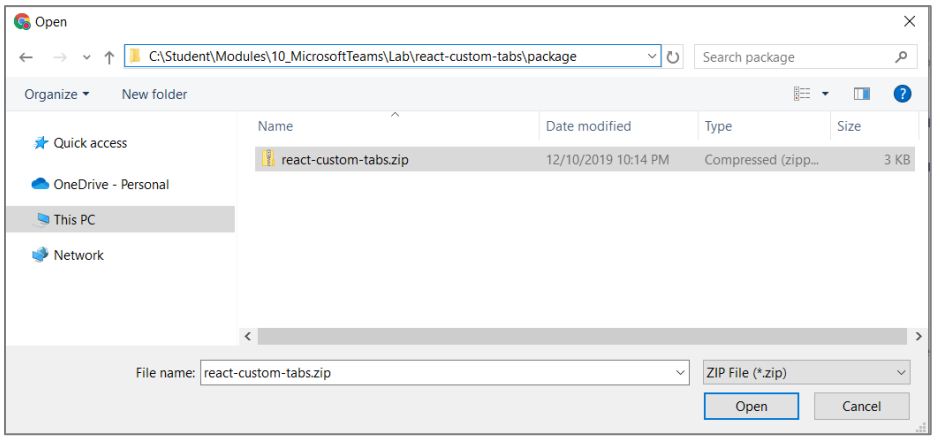


d) Locate and click the **Upload a custom app** link in the bottom left corner.

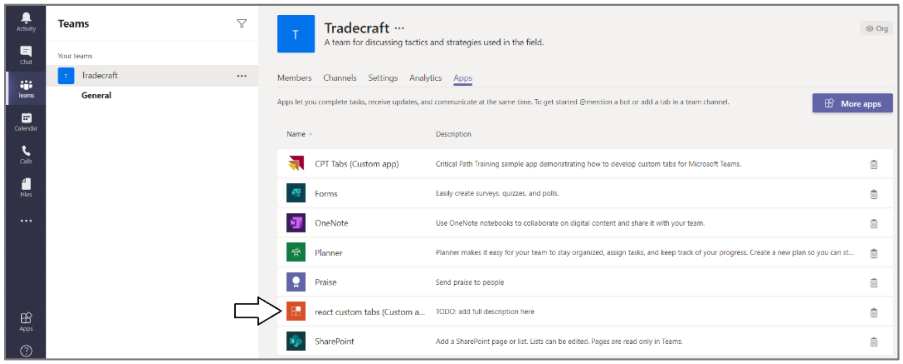
e) Upload the same app package you uploaded in a previous step named **CptTeamsTabsApp.zip**.

f) Upload the app package from the following path.

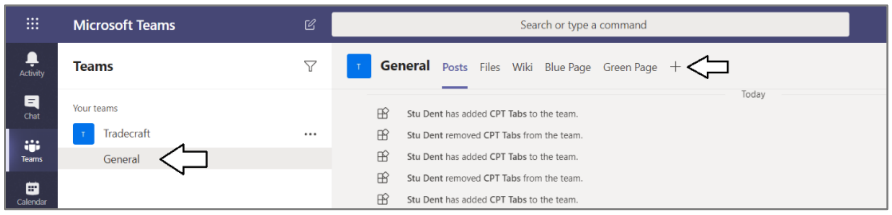
```
C:\Student\Modules\10_MicrosoftTeams\Lab\react-custom-tabs\package
```



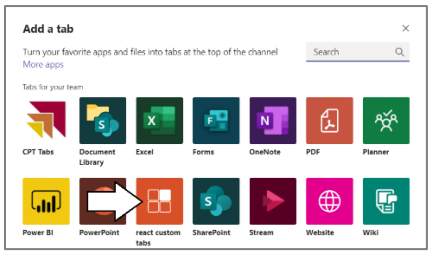
g) You should now see the **react custom tabs** app has been installed in the scope of the **Tradecraft** team.



h) Navogate to the General tab and click **+** to add a new tab.

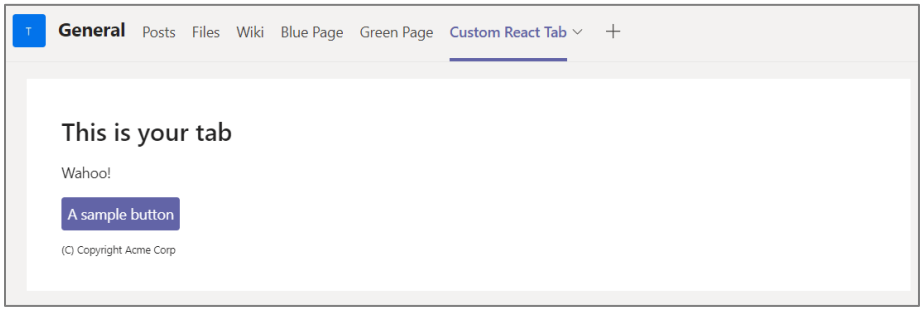


i) Select the **react custom tabs**.

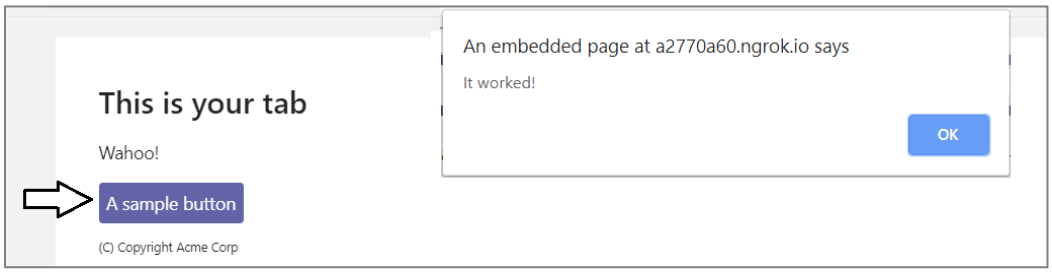


j) Add some text into the **enter a value** textbox and click **Save**.

k) You have now create a new dynamic tab using the **react-custom-tabs** project.



l) Click the button to see what it does.



Exercise 4: Working on Additional Teams Lab Exercise provided by Microsoft

If you are looking for more lab work, try out these labs provided by the Microsoft product team.

1. Try this lab to Create and Test a Microsoft Teams app using Yeoman

   https://github.com/OfficeDev/TrainingContent/blob/master/Teams/04%20Fundamentals%20of%20Microsoft%20Teams/Lab.md

2. Try this lab to Create and test a basic Microsoft Teams bot using Visual Studio

   https://github.com/OfficeDev/TrainingContent/blob/master/Teams/04%20Fundamentals%20of%20Microsoft%20Teams/Lab.md#exercise2

3. Try this lab to Call the Microsoft Graph API inside a tab.

   https://github.com/OfficeDev/TrainingContent/blob/master/Teams/04%20Fundamentals%20of%20Microsoft%20Teams/Lab.md#exercise3