Lab 5: Connect web services to Microsoft Teams with webhooks and Office 365 Connectors

Contents

[Exercise 1 - Create outgoing webhooks 2](#_Toc33100554)

[Create Microsoft Teams app 2](#_Toc33100555)

[Code the outgoing webhook 3](#_Toc33100556)

[Test the outgoing webhook 11](#_Toc33100557)

[Summary 15](#_Toc33100558)

[Exercise 2- Create incoming webhooks 16](#_Toc33100559)

[Register a new incoming webhook 16](#_Toc33100560)

[Test the incoming webhook 19](#_Toc33100561)

[Summary 22](#_Toc33100562)

[Exercise 3 - Create and add Office 365 Connectors to teams 23](#_Toc33100563)

[Register an Office 365 Connector 23](#_Toc33100564)

[Create Microsoft Teams app 24](#_Toc33100565)

[Examine and update the app manifest file 25](#_Toc33100566)

[Examine the configuration page 26](#_Toc33100567)

[Examine the web service 27](#_Toc33100568)

[Test the Office 365 Connector in Microsoft Teams 28](#_Toc33100569)

[Summary 36](#_Toc33100570)

# Exercise 1 - Create outgoing webhooks

In this exercise, you’ll learn how to create a web service and register it as an outgoing webhook in Microsoft Teams.

## Create Microsoft Teams app

Open your command prompt, navigate to a directory where you want to save your work, create a new folder **learn-msteams-taskmodules**, and change directory into that folder.

Run the Yeoman Generator for Microsoft Teams by running the following command:

yo teams



Yeoman will launch and ask you a series of questions. Answer the questions with the following values:

* **What is your solution name?**: TeamsWebhooks
* **Where do you want to place the files?**: Use the current folder
* **Title of your Microsoft Teams App project?**: Teams Webhooks
* **Your (company) name? (max 32 characters)**: Contoso
* **Which manifest version would you like to use?**: 1.5
* **Enter your Microsoft Partner Id, if you have one?**: (Leave blank to skip)
* **What features do you want to add to your project?**: An Outgoing Webhook
* **The URL where you will host this solution?**: [https://teamswebhooks.azurewebsites.net](https://teamswebhooks.azurewebsites.net/)
* **Would you like to include Test framework and initial tests?**: No
* **Would you like to use Azure Applications Insights for telemetry?**: No
* **Name of your outgoing webhook?**: Teams Webhooks Outgoing Webhook

**Note**

Most of the answers to these questions can be changed after creating the project. For example, the URL where the project will be hosted isn't important at the time of creating or testing the project.

After answering the generator's questions, the generator will create the scaffolding for the project and then execute npm install that downloads all the dependencies required by the project.

Our web service will need one more NPM package to simplify finding data in an array. Execute the following command in the command prompt from the root folder of the project to install the library Lodash:

npm install lodash -S

## Code the outgoing webhook

The Yeoman Generator for Microsoft Teams created a stub web service endpoint for our outgoing webhook. Locate and open the file **./src/app/teamsWebhooksOutgoingWebhook/TeamsWebhooksOutgoingWebhook.ts**. It listens for HTTPS requests at the endpoint **/api/webhook**.

Find the requestHander() method in the TeamsWebhooksOutgoingWebhook class. The method first checks the HMAC value in the authorization header against the security token that you'll obtain when you add the webhook to a team.

Locate the following code:

message.text = `Echo ${incoming.text}`;

This code simply echoes the string entered in the message back to Microsoft Teams that will be added in a reply to the message that triggered the webhook.

Let's update this code to add some real functionality. When a user @mentions the bot, if they enter the name of a known planet of our solar system, it will respond with an adaptive card that displays details of the planet.

Let's start by adding two resource files to the project.

Create a new file **planets.json** in the **./src/app/teamsWebhooksOutgoingWebhook** folder and add the following JSON to it. This file will contain an array of planet details:

[

{

"id": "1",

"name": "Mercury",

"summary": "Mercury is the smallest and innermost planet in the Solar System. Its orbit around the Sun takes 87.97 days, the shortest of all the planets in the Solar System. It is named after the Roman deity Mercury, the messenger of the gods.",

"solarOrbitYears": 0.24,

"solarOrbitAvgDistanceKm": 57909050,

"numSatellites": 0,

"wikiLink": "https://en.wikipedia.org/wiki/Mercury\_(planet)",

"imageLink": "https://upload.wikimedia.org/wikipedia/commons/d/d9/Mercury\_in\_color\_-\_Prockter07-edit1.jpg",

"imageAlt": "NASA/Johns Hopkins University Applied Physics Laboratory/Carnegie Institution of Washington [Public domain]"

},

{

"id": "2",

"name": "Venus",

"summary": "Venus is the second planet from the Sun. It is named after the Roman goddess of love and beauty. As the second-brightest natural object in the night sky after the Moon, Venus can cast shadows and, rarely, is visible to the naked eye in broad daylight. Venus lies within Earth's orbit, and so never appears to venture far from the Sun, setting in the west just after dusk and rising in the east a bit before dawn.",

"solarOrbitYears": 0.62,

"solarOrbitAvgDistanceKm": 108208000,

"numSatellites": 0,

"wikiLink": "https://en.wikipedia.org/wiki/Venus",

"imageLink": "https://upload.wikimedia.org/wikipedia/commons/e/e5/Venus-real\_color.jpg",

"imageAlt": "&quot;Image processing by R. Nunes&quot;, link to http://www.astrosurf.com/nunes [Public domain]"

},

{

"id": "3",

"name": "Earth",

"summary": "Earth is the third planet from the Sun and the only astronomical object known to harbor life. According to radiometric dating and other sources of evidence, Earth formed over 4.5 billion years ago. Earth's gravity interacts with other objects in space, especially the Sun and the Moon, which is Earth's only natural satellite. Earth orbits around the Sun in 365.256 days, a period known as an Earth sidereal year. During this time, Earth rotates about its axis about 366.256 times.",

"solarOrbitYears": 1.00,

"solarOrbitAvgDistanceKm": 149597500,

"numSatellites": 1,

"wikiLink": "https://en.wikipedia.org/wiki/Earth",

"imageLink": "https://upload.wikimedia.org/wikipedia/commons/9/97/The\_Earth\_seen\_from\_Apollo\_17.jpg",

"imageAlt": "Apollo 17 [Public domain]"

},

{

"id": "4",

"name": "Mars",

"summary": "Mars is the fourth planet from the Sun and the second-smallest planet in the Solar System after Mercury. In English, Mars carries a name of the Roman god of war and is often referred to as the 'Red Planet'. The latter refers to the effect of the iron oxide prevalent on Mars' surface, which gives it a reddish appearance distinctive among the astronomical bodies visible to the naked eye.",

"solarOrbitYears": 1.88,

"solarOrbitAvgDistanceKm": 134935000,

"numSatellites": 2,

"wikiLink": "https://en.wikipedia.org/wiki/Mars",

"imageLink": "https://upload.wikimedia.org/wikipedia/commons/0/02/OSIRIS\_Mars\_true\_color.jpg",

"imageAlt": "ESA - European Space Agency &amp; Max-Planck Institute for Solar System Research for OSIRIS Team ESA/MPS/UPD/LAM/IAA/RSSD/INTA/UPM/DASP/IDA [CC BY-SA 3.0-IGO (https://creativecommons.org/licenses/by-sa/3.0-igo)]"

},

{

"id": "5",

"name": "Jupiter",

"summary": "Jupiter is the fifth planet from the Sun and the largest in the Solar System. It is a gas giant with a mass one-thousandth that of the Sun, but two-and-a-half times that of all the other planets in the Solar System combined. Jupiter is one of the brightest objects visible to the naked eye in the night sky, and has been known to ancient civilizations since before recorded history. It is named after the Roman god Jupiter. When viewed from Earth, Jupiter can be bright enough for its reflected light to cast shadows, and is on average the third-brightest natural object in the night sky after the Moon and Venus.",

"solarOrbitYears": 11.86,

"solarOrbitAvgDistanceKm": 445336000,

"numSatellites": 78,

"wikiLink": "https://en.wikipedia.org/wiki/Jupiter",

"imageLink": "https://upload.wikimedia.org/wikipedia/commons/5/50/Jupiter%2C\_image\_taken\_by\_NASA%27s\_Hubble\_Space\_Telescope%2C\_June\_2019\_-\_Edited.jpg",

"imageAlt": "NASA, ESA, and A. Simon (NASA Goddard), edited by PlanetUser [Public domain]"

},

{

"id": "6",

"name": "Saturn",

"summary": "Saturn is the sixth planet from the Sun and the second-largest in the Solar System, after Jupiter. It is a gas giant with an average radius about nine times that of Earth. It has only one-eighth the average density of Earth; however, with its larger volume, Saturn is over 95 times more massive. Saturn is named after the Roman god of wealth and agriculture; its astronomical symbol (♄) represents the god's sickle.",

"solarOrbitYears": 29.46,

"solarOrbitAvgDistanceKm": 1433525000,

"numSatellites": 82,

"wikiLink": "https://en.wikipedia.org/wiki/Saturn",

"imageLink": "https://upload.wikimedia.org/wikipedia/commons/c/c7/Saturn\_during\_Equinox.jpg",

"imageAlt": "NASA / JPL / Space Science Institute [Public domain]"

},

{

"id": "7",

"name": "Uranus",

"summary": "Uranus is the seventh planet from the Sun. It has the third-largest planetary radius and fourth-largest planetary mass in the Solar System. Uranus is similar in composition to Neptune, and both have bulk chemical compositions which differ from that of the larger gas giants Jupiter and Saturn. For this reason, scientists often classify Uranus and Neptune as \"ice giants\" to distinguish them from the gas giants.",

"solarOrbitYears": 84.02,

"solarOrbitAvgDistanceKm": 2883000000,

"numSatellites": 27,

"wikiLink": "https://en.wikipedia.org/wiki/Uranus",

"imageLink": "https://upload.wikimedia.org/wikipedia/commons/3/3d/Uranus2.jpg",

"imageAlt": "NASA/JPL-Caltech [Public domain]"

},

{

"id": "8",

"name": "Neptune",

"summary": "Neptune is the eighth and farthest known planet from the Sun in the Solar System. In the Solar System, it is the fourth-largest planet by diameter, the third-most-massive planet, and the densest giant planet. Neptune is 17 times the mass of Earth, slightly more massive than its near-twin Uranus. Neptune is denser and physically smaller than Uranus because its greater mass causes more gravitational compression of its atmosphere.",

"solarOrbitYears": 164.80,

"solarOrbitAvgDistanceKm": 4500000000,

"numSatellites": 14,

"wikiLink": "https://en.wikipedia.org/wiki/Neptune",

"imageLink": "https://upload.wikimedia.org/wikipedia/commons/6/63/Neptune\_-\_Voyager\_2\_%2829347980845%29\_flatten\_crop.jpg",

"imageAlt": "Justin Cowart [CC BY (https://creativecommons.org/licenses/by/2.0)]"

}

]

Next, create a new file **planetDisplayCard.json** in the **./src/app/teamsWebhooksOutgoingWebhook** folder and add the following JSON to it. This file will contain a template of the adaptive card the web service will respond with:

{

"$schema": "http://adaptivecards.io/schemas/adaptive-card.json",

"type": "AdaptiveCard",

"version": "1.0",

"body": [

{

"id": "cardHeader",

"type": "Container",

"items": [

{

"id": "planetName",

"type": "TextBlock",

"weight": "bolder",

"size": "medium"

}

]

},

{

"type": "Container",

"id": "cardBody",

"items": [

{

"id": "planetSummary",

"type": "TextBlock",

"wrap": true

},

{

"id": "planetDetails",

"type": "ColumnSet",

"columns": [

{

"type": "Column",

"width": "100",

"items": [

{

"id": "planetImage",

"size": "stretch",

"type": "Image"

}

]

},

{

"type": "Column",

"width": "250",

"items": [

{

"type": "FactSet",

"facts": [

{

"id": "orderFromSun",

"title": "Order from the sun:"

},

{

"id": "planetNumSatellites",

"title": "Known satellites:"

},

{

"id": "solarOrbitYears",

"title": "Solar orbit (\*Earth years\*):"

},

{

"id": "solarOrbitAvgDistanceKm",

"title": "Average distance from the sun (\*km\*):"

}

]

}

]

}

]

},

{

"id": "imageAttribution",

"type": "TextBlock",

"size": "medium",

"isSubtle": true,

"wrap": true

}

]

}

],

"actions": [

{

"type": "Action.OpenUrl",

"title": "Learn more on Wikipedia"

}

]

}

Add the following import statement to the **./src/app/teamsWebhooksOutgoingWebhook/TeamsWebhooksOutgoingWebhook.ts** file, just after the existing import statements:

import { find, sortBy } from "lodash";

Add the following method to the TeamsWebhooksOutgoingWebhook class. The getPlanetDetailCard() method will load and populate the adaptive card template with details using the provided planet object:

private static getPlanetDetailCard(selectedPlanet: any): builder.Attachment {

// load display card

const adaptiveCardSource: any = require("./planetDisplayCard.json");

// update planet fields in display card

adaptiveCardSource.actions[0].url = selectedPlanet.wikiLink;

find(adaptiveCardSource.body, { "id": "cardHeader" }).items[0].text = selectedPlanet.name;

const cardBody: any = find(adaptiveCardSource.body, { "id": "cardBody" });

find(cardBody.items, { "id": "planetSummary" }).text = selectedPlanet.summary;

find(cardBody.items, { "id": "imageAttribution" }).text = "\*Image attribution: " + selectedPlanet.imageAlt + "\*";

const cardDetails: any = find(cardBody.items, { "id": "planetDetails" });

cardDetails.columns[0].items[0].url = selectedPlanet.imageLink;

find(cardDetails.columns[1].items[0].facts, { "id": "orderFromSun" }).value = selectedPlanet.id;

find(cardDetails.columns[1].items[0].facts, { "id": "planetNumSatellites" }).value = selectedPlanet.numSatellites;

find(cardDetails.columns[1].items[0].facts, { "id": "solarOrbitYears" }).value = selectedPlanet.solarOrbitYears;

find(cardDetails.columns[1].items[0].facts, { "id": "solarOrbitAvgDistanceKm" }).value = Number(selectedPlanet.solarOrbitAvgDistanceKm).toLocaleString();

// return the adaptive card

return builder.CardFactory.adaptiveCard(adaptiveCardSource);

}

Next, add the following method ot the TeamsWebhooksOutgoingWebhook class. The processAuthenticatedRequest() method takes the incoming text uses it to find a planet in the **planets.json** file. If it finds one, it calls the getPlanetDetailCard() method to get an adaptive card and returns it as an Activity that will be sent back to Microsoft Teams. If a planet isn't found, it just echoes the text back in a reply to the request:

private static processAuthenticatedRequest(incomingText: string): Partial<builder.Activity> {

const message: Partial<builder.Activity> = {

type: builder.ActivityTypes.Message

};

// load planets

const planets: any = require("./planets.json");

// get the selected planet

const selectedPlanet: any = planets.filter((planet) => (planet.name as string).trim().toLowerCase() === incomingText.trim().toLowerCase());

if (!selectedPlanet || !selectedPlanet.length) {

message.text = `Echo ${incomingText}`;

} else {

const adaptiveCard = TeamsWebhooksOutgoingWebhook.getPlanetDetailCard(selectedPlanet[0]);

message.type = "result";

message.attachmentLayout = "list";

message.attachments = [adaptiveCard];

}

return message;

}

Add the following scrubMessage() method to the TeamsWebhooksOutgoingWebhook class. A user must @mention an outgoing webhook in order to send a message to it. This method will remove the <at></at> text and any spaces to extract the planet name:

private static scrubMessage(incomingText: string): string {

let cleanMessage = incomingText

.slice(incomingText.lastIndexOf(">")+1, incomingText.length)

.replace("&nbsp;", "");

return cleanMessage;

}

Finally, update the requestHandler() method:

* Locate the following code and change the message declaration from a const to let as you will change change this value.

const message: Partial<builder.Activity> = {

type: builder.ActivityTypes.Message

};

...

* Locate and replace the following code:

message.text = `Echo ${incoming.text}`;

with the following code:

const scrubbedText = TeamsWebhooksOutgoingWebhook.scrubMessage(incoming.text)

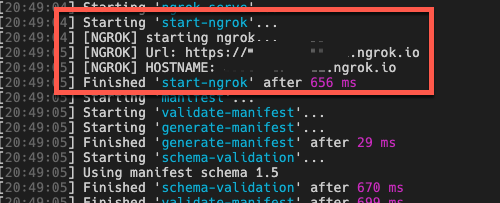
message = TeamsWebhooksOutgoingWebhook.processAuthenticatedRequest(scrubbedText);

## Test the outgoing webhook

From the command line, navigate to the root folder for the project and execute the following command:

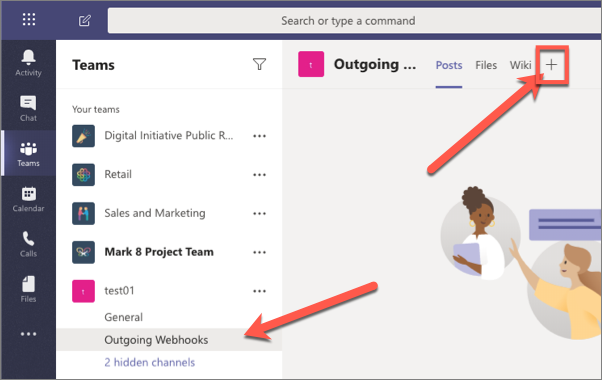
gulp ngrok-serve

In the console, locate the dynamic URL created by ngrok:

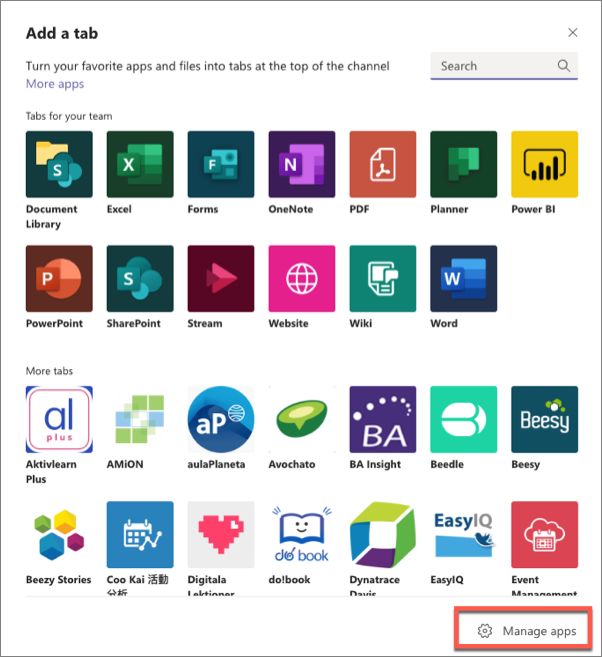


Now let's add the outgoing webhook to a team in Microsoft Teams. In the browser, navigate to [**https://teams.microsoft.com**](https://teams.microsoft.com/) and sign in with the credentials of a Work and School account.

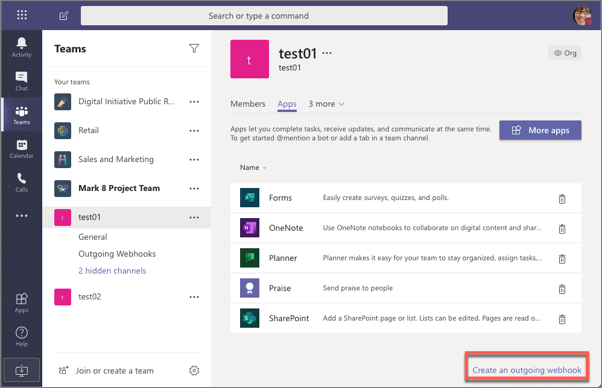
Once you are signed in, select a channel in a team you want to add the webhook to. From the channel's page, select the **+** in the top navigation:



On the **Add a tab** dialog, select **Manage apps** in the lower right corner:



This will take you to the **Manage Channel** page. Select the **Create an outgoing webhook** in the lower right corner:



In the **Create an outgoing webhook** dialog, enter the following values and select **Create**:

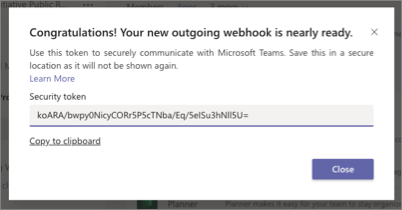
* **Name**: Planet Details
* **Callback URL**: https://{{REPLACE\_NGROK\_SUBDOMAIN}}.ngrok.io/api/webhook

**Important**

Replace the {{REPLACE\_NGROK\_SUBDOMAIN}} with the value of your dynamically created Ngrok URL displayed in the console when you ran gulp ngrok-serve.

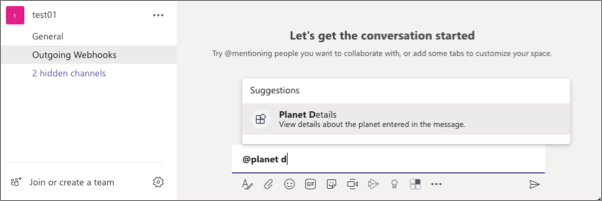
* **Description**: View details about the planet entered in the message.

After creating the outgoing webhook, Microsoft Teams will display a security token.

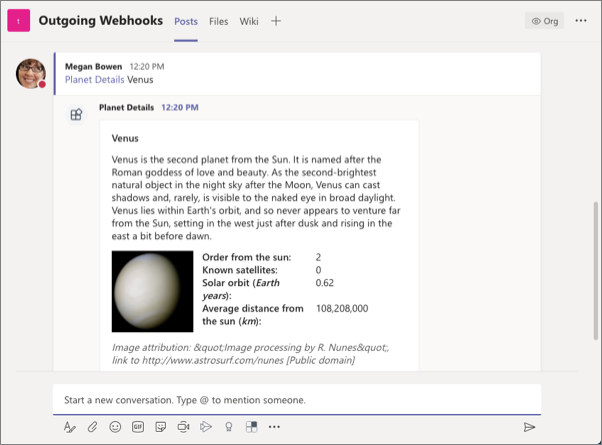


Copy this value and and set the SECURITY\_TOKEN property in the **./.env** file in the project.

Now you can test the webhook. Go to a channel's **Conversation** tab within the team and enter the message **@Planet Details Venus**. Notice that as you're typing the message, Microsoft Teams detects the name of the webhook:



A few seconds after submitting the message, you'll see a reply to your message appear that contains the customized adaptive card with details about the planet:



You've successfully tested your outgoing webhook! Stop the local web server by pressing CTRL+C in the command prompt.

## Summary

In this exercise, you learned how to create a web service and register it as an outgoing webhook in Microsoft Teams.

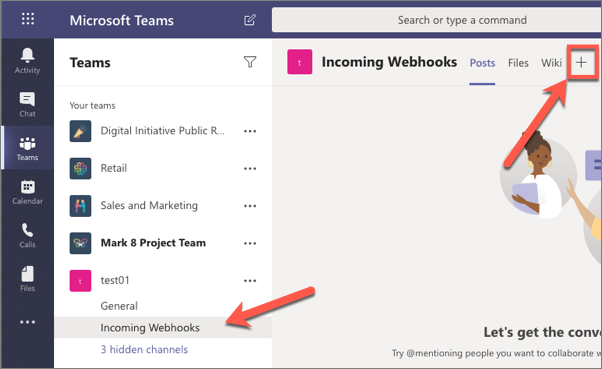
# Exercise 2- Create incoming webhooks

In this exercise, you’ll learn how to register an incoming webhook in a Microsoft Teams channel and post a message to it.

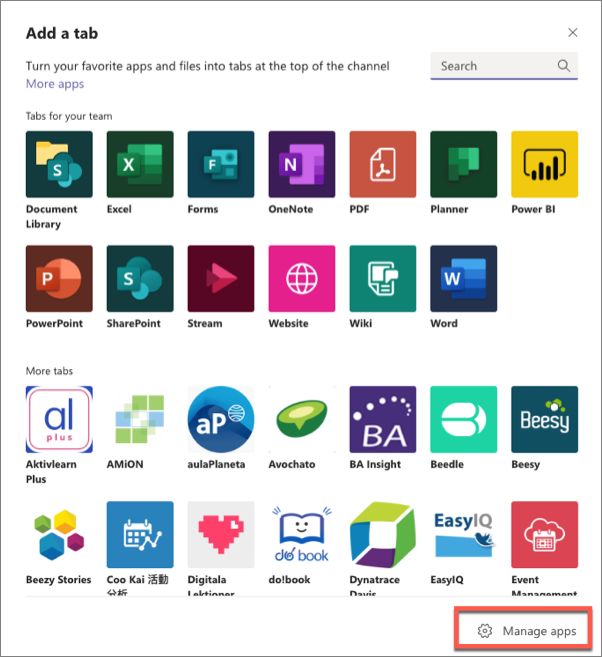
## Register a new incoming webhook

Now let's add the outgoing webhook to a team in Microsoft Teams. In the browser, navigate to [**https://teams.microsoft.com**](https://teams.microsoft.com/) and sign in with the credentials of a Work and School account.

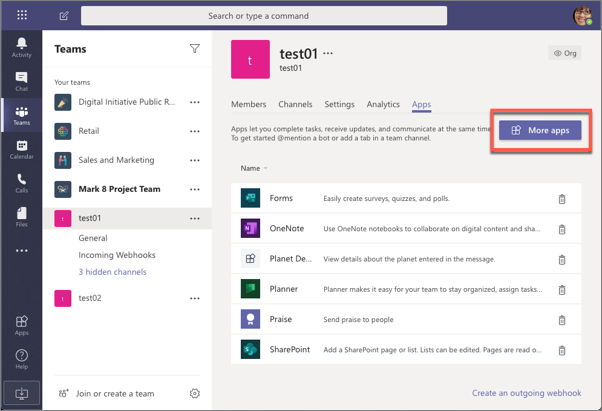
Once you are signed in, select a channel in a team you want to add the webhook to. From the channel's page, select the **+** in the top navigation:



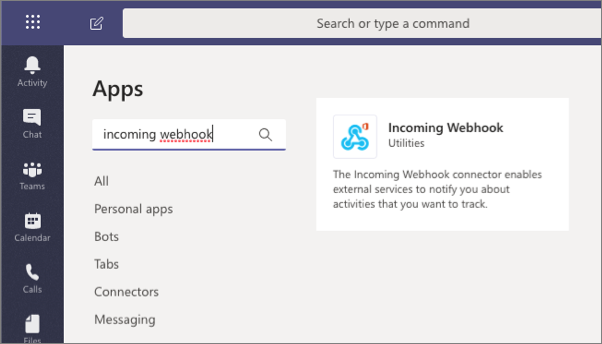
On the **Add a tab** dialog, select **Manage apps** in the lower right corner:



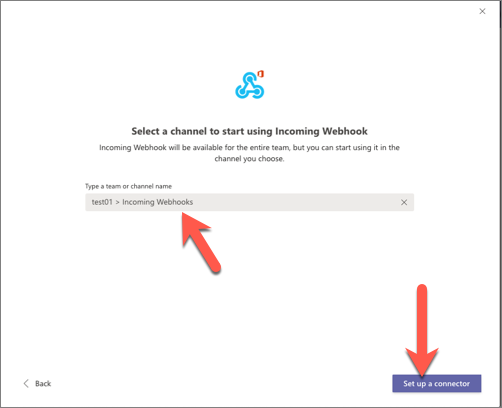
This will take you to the **Manage Channel** page. Select the **More apps** button:



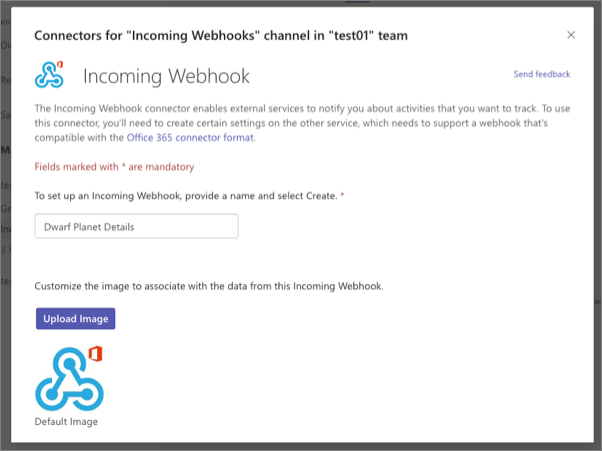
On the **Apps** page, search for the **incoming webhook** app and select it.



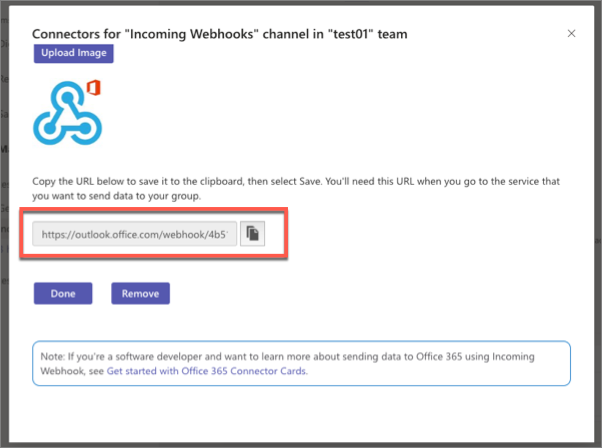
Select the **Add to a team** button. On the next page, enter the name of a channel to add the webhook to, and select the **Set up a connector**:



On the **Incoming Webhook** configuration screen, enter the name **Dwarf Planet Details** and select the **Create** button (*you may need to scroll down in the dialog as the****Create****button isn't visible by default, as in the following screenshot*):



After creating the incoming webhook, the dialog will add a new input box that contains the endpoint for you should post to. Copy this value:

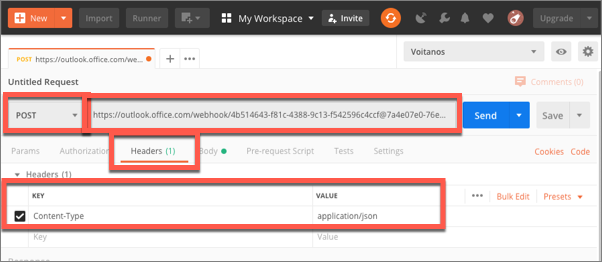


## Test the incoming webhook

After configuring the incoming webhook, the next step is to submit a post to it to display a message in the channel. Do this by submitting an HTTPS request to the webhook endpoint provided.

Using the free tool [Postman](https://www.postman.com/), create a new request to the point endpoint:

* set the request to a **POST**
* set the endpoint to the webhook endpoint you copied at the end of the previous section
* set the Content-Type header to application/json on the **Headers** tab:



* add the following JSON to the **Body** tab and select the **raw** option:

{

"@type": "MessageCard",

"@context": "http://schema.org/extensions",

"summary": "Dwarf planet Pluto details",

"sections": [{

"activityTitle": "Dwarf planet Pluto details",

"activityImage": "https://upload.wikimedia.org/wikipedia/commons/e/ef/Pluto\_in\_True\_Color\_-\_High-Res.jpg",

"facts": [

{

"name": "Description",

"value": "Pluto is an icy dwarf planet in the Kuiper belt, a ring of bodies beyond the orbit of Neptune. It was the first Kuiper belt object to be discovered and is the largest known dwarf planet. Pluto was discovered by Clyde Tombaugh in 1930 as the ninth planet from the Sun. After 1992, its status as a planet was questioned following the discovery of several objects of similar size in the Kuiper belt. In 2005, Eris, a dwarf planet in the scattered disc which is 27% more massive than Pluto, was discovered. This led the International Astronomical Union (IAU) to define the term \"planet\" formally in 2006, during their 26th General Assembly. That definition excluded Pluto and reclassified it as a dwarf planet."

},

{

"name": "Order from the sun",

"value": "9"

},

{

"name": "Known satellites",

"value": "5"

},

{

"name": "Solar orbit (\*Earth years\*)",

"value": "247.9"

},

{

"name": "Average distance from the sun (\*km\*)",

"value": "590637500000"

},

{

"name": "Image attribution",

"value": "NASA/Johns Hopkins University Applied Physics Laboratory/Southwest Research Institute/Alex Parker [Public domain]"

}

]

}],

"potentialAction": [{

"@context": "http://schema.org",

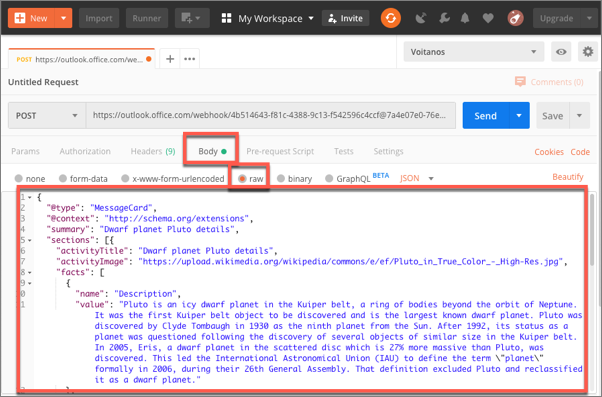
"@type": "ViewAction",

"name": "Learn more on Wikipedia",

"target": ["https://en.wikipedia.org/wiki/Pluto"]

}]

}

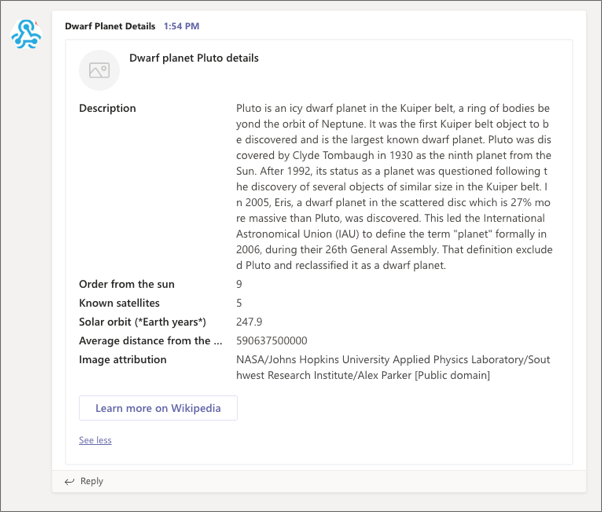


**Important**

If you include a card in a message sent to an incoming webhook, it must be an Office 365 Connector Cards; adaptive cards aren't supported when sending messages to incoming webhooks.

Send the card to Microsoft Teams by selecting the **Send** button.

In the browser, navigate back to the **Conversations** tab in the channel where you installed the incoming webhook. You should see a message containing your card:



## Summary

In this exercise, you learned how to register an incoming webhook in a Microsoft Teams channel and post a message to it.

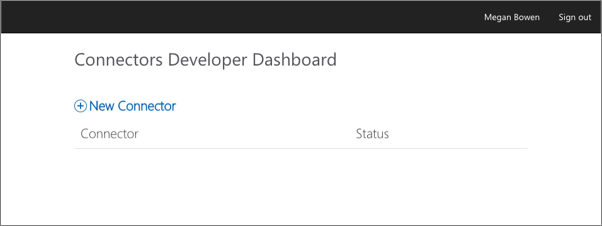
# Exercise 3 - Create and add Office 365 Connectors to teams

In this unit, you’ll learn how to create an Office 365 Connector and add it to Microsoft Teams.

The first step will be to register a new Office 365 Connector with the **Connectors Developer Dashboard**. Then you'll create a new Microsoft Teams app that contains a web service and the necessary details to associate the Office 365 Connector to connect it to Microsoft Teams.

## Register an Office 365 Connector

Open a browser and navigate to the **Connectors Developer Dashboard**: <https://aka.ms/ConnectorsDashboard>



Select **New Connector**.

On the **Register Connector** page, complete the required fields in the form with anything you like and accept any default options presented, with the following exceptions:

* **Connector name**: My First Teams Connector
* **Configuration page for your Connector**: [https://REPLACE.ngrok.io/MyFirstTeamsConnector/config.html](https://replace.ngrok.io/MyFirstTeamsConnector/config.html)

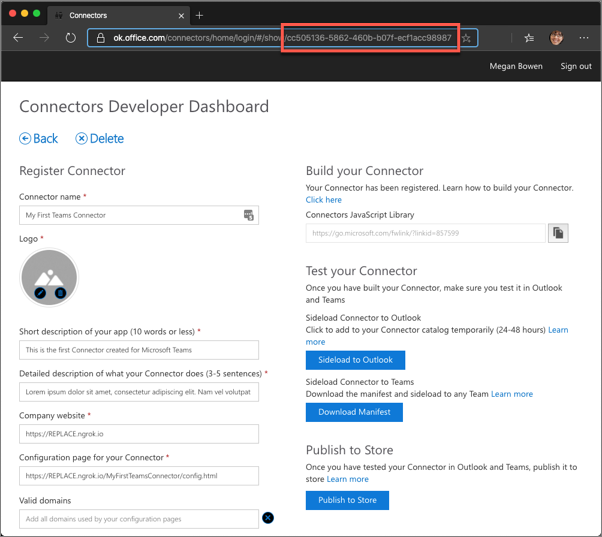
**Note**

In this exercise, it isn't necessary to come back and update the address of the configuration page because we are only testing the Connector in Microsoft Teams.

Select the **I accept the terms and conditions...** checkbox and select **Save** to register the Connector.

After successfully registering your Connector, the **Connectors Developer Dashboard** page will display some additional sections. While there is a button to **Download Manifest** for a custom Microsoft Teams app, we'll use the manifest created by the Yeoman Generator for Microsoft Teams.

You'll need the ID of your new Connector later in the exercise. This ID, a GUID, can be found in the URL of the updated page. Copy this ID for later use.



## Create Microsoft Teams app

Open your command prompt, navigate to a directory where you want to save your work, create a new folder **learn-msteams-taskmodules**, and change directory into that folder.

Run the Yeoman Generator for Microsoft Teams by running the following command:

yo teams

Yeoman will launch and ask you a series of questions. Answer the questions with the following values:

* **What is your solution name?**: MyFirstTeamsConnector
* **Where do you want to place the files?**: Use the current folder
* **Title of your Microsoft Teams App project?**: My First Teams Connector
* **Your (company) name? (max 32 characters)**: Contoso
* **Which manifest version would you like to use?**: 1.5
* **Enter your Microsoft Partner Id, if you have one?**: (Leave blank to skip)
* **What features do you want to add to your project?**: A Connector
* **The URL where you will host this solution?**: [https://myfirstteamsconnector.azurewebsites.net](https://myfirstteamsconnector.azurewebsites.net/)
* **Would you like to include Test framework and initial tests?**: No
* **Would you like to use Azure Applications Insights for telemetry?**: No
* **What type of Connector would you like to include?**: A new Connector hosted in this solution
* **What is the Id of your Connector (found in the Connector Developer Portal)?**: (Enter the ID of the Connector you copied in the last step)
* **What is the name of your Connector?** My First Teams Connector

**Note**

Most of the answers to these questions can be changed after creating the project. For example, the URL where the project will be hosted isn't important at the time of creating or testing the project.

After answering the generator's questions, the generator will create the scaffolding for the project and then execute npm install that downloads all the dependencies required by the project.

## Examine and update the app manifest file

After creating the project you'll need to make a few edits to the default app manifest file. Locate and open the file **./src/manifest/manifest.json**.

Within this file, locate the connectors array. Notice a single connector is listed:

"connectors": [

{

"connectorId": "{{CONNECTOR\_ID}}",

"configurationUrl": "https://{{HOSTNAME}}/myFirstTeamsConnector/config.html",

"scopes": [

"team"

]

}

]

**Important**

Notice the scope of the Connector ID will be replaced by the build process. This ID can be found in the ./.env file that's used for development.

A default manifest has a few empty array properties must be removed in order to add the Connector to a team. Locate the following properties and delete them from the **manifest.json** file:

"configurableTabs": []

"staticTabs": []

"bots": []

"composeExtensions": []

Finally, let's explore the code in the default project to see how it works. The default project contains two elements that support the Connector:

1. The web service that Microsoft Teams & the Office 365 Connector infrastructure will call which implements the Connector logic.
2. A configuration page that's displayed when the Connector is added to a team.

## Examine the configuration page

The configuration page is an HTML page that contains a React control. Locate and open the React control file **./src/app/scripts/myFirstTeamsConnector/MyFirstTeamsConnectorConfig.tsx**.

You need to make one edit to this file before building the project. Within the componentWillMount() React lifecycle event handler, locate the following code:

microsoftTeams.getContext((context: microsoftTeams.Context) => {

this.setState({

color: availableColors.find(c => c.code === context.entityId),

});

this.setValidityState(this.state.color !== undefined);

});

The find() method needs to be replaced with a fitler() method. Update this line:

color: availableColors.find(c => c.code === context.entityId),

... to the following:

color: availableColors.filter(c => c.code === context.entityId)[0],

The important part of this component to take note of is in the call to the microsoftTeams.settings.registerOnSaveHandler() handler that is called when the user selects the **Save** button on the config page. Selecting **Save** will save the configuration of the Connector in Microsoft Teams and notify the Connector's web service that it has been added to a team.

The code does the following things:

* Update the settings for the Connector in Microsoft Teams
* Submit an HTTP POST to the Connector's /api/connector/connect endpoint with a payload that contains necessary information the Connector web service needs to store

Notice before the HTTP POST is executed, the code is calls the getSettings() method to retrieve the settings from Microsoft Teams:

microsoftTeams.settings.getSettings((s: any) => {

this.setState({

webhookUrl: s.webhookUrl,

user: s.userObjectId,

appType: s.appType,

});

...

This method passes in a settings object that contains a few important properties:

* **webhookUrl**: this is the endpoint, the incoming webhook, the Connector will submit to
* **userObjectId**: the ID of the user who is registering the Connector
* **appType**: this is the value team when associating the Connector with a team

When the HTTP POST request succeeds, the notifySuccess() method is called on the SaveEvent passed into this registerOnSaveHandler() handler; otherwise the notifyFailure() method is called.

## Examine the web service

The Connector is implemented as a web service that is hosted within the Microsoft Teams project. Locate and open the file **./src/app/myFirstTeamsConnector/MyFirstTeamsConnector.ts**.

The web service exposes two endpoints, both represented by methods with the endpoint's respective names. The endpoints are:

* **/connect**: This is called, as we covered in the previous section, when the **Save** button is selected on the config page. The default code in the handler, Connect(), simply saves the Connector registration in a local JSON file.

**Note**

The sample web service in the default project uses a JSON file to store the registration. It does not contain any logic to update an existing Connector when it's changed, or remove a Connector. In a real world Connector, you'll likely implement a system that saves this registration to a persistent data store that handles the scenarios of updating and removing a Connector from a team.

* **/ping**: This endpoint can be called by anyone and is simply used to test the Connector. When called, it will create a card and send it to all the registered Connectors.

## Test the Office 365 Connector in Microsoft Teams

At this point, our Microsoft Teams app and Office 365 Connector is set up and working. Verify this by starting ngrok and install the Connector.

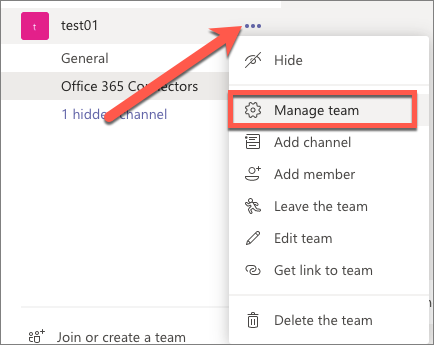
From the command line, navigate to the root folder for the project and execute the following command:

gulp ngrok-serve

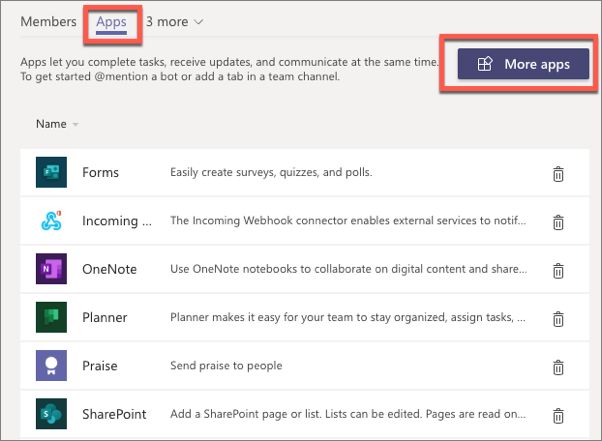
Now let's load the Connector in Microsoft Teams. In the browser, navigate to [**https://teams.microsoft.com**](https://teams.microsoft.com/) and sign in with the credentials of a Work and School account.

Once you are signed in, the first step is to install the Microsoft Teams app.

Select a team, select the action menu on the team and select **Manage team**:

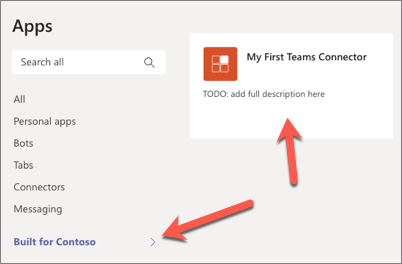


Select the **Apps** tab and then the **More apps** button:

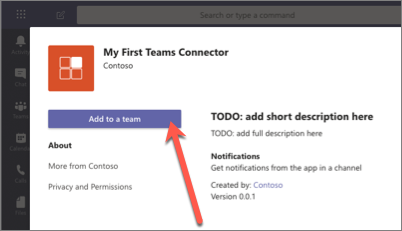


From the **Browse available apps and services**, select the **Upload a custom app > Upload for [tenant]** at the bottom of the **Apps** panel of categories. Select the Microsoft Teams app package, the **MyFirstTeamsConnector.zip** file in the **./package** folder of your project.

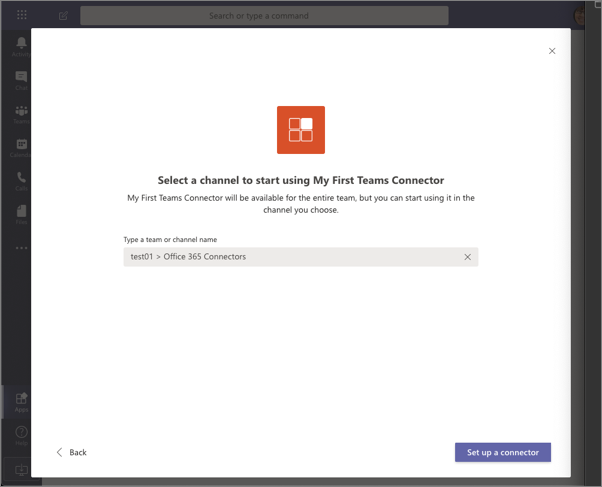
After uploading the app, Microsoft Teams will display it on the list of apps installed under the **Build for [tenant]** category page:



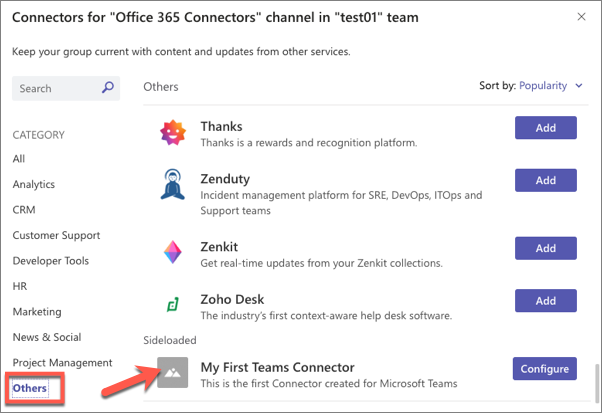
Once installed, you can now add the Connector to a team. You can do this from the app by selecting it, then select the **Add to a team** button and enter the team...



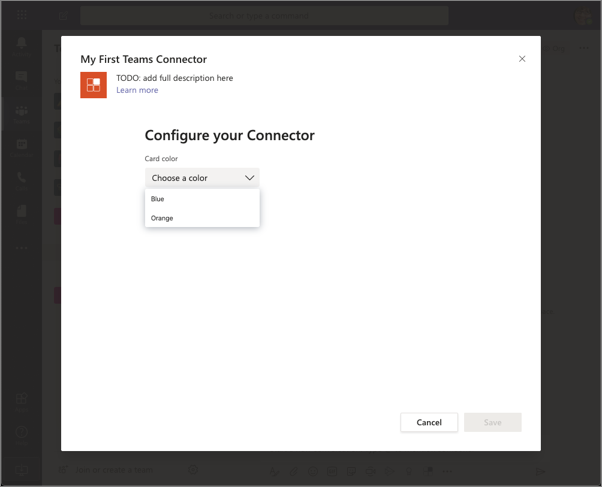
... or you can add it directly to a team. Let's use this first option. Select the **Add to a team** button and select a team to add the Connector to:



On the **Connectors for... channel in ... team** page, select the **Others** category and scroll to the bottom. You will see your connector under the **Sideloaded** section:



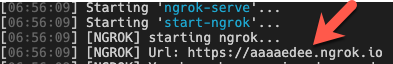
Select the **Configure** button for the Connector. This will display the configuration page from our project:



Select a color and then select the **Save** button.

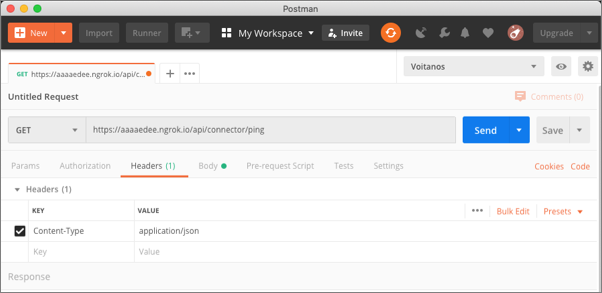
This will trigger the configuration page to call the web service's /connect endpoint to save the connector.

With the Connector saved, the next step is to see it post to a channel. Do this by submitting an HTTPS request to the web service's /ping endpoint. To do this, you need to know the dynamic URL created by the ngrok utility. You can retrieve this from the command prompt as it was displayed when you started the gulp ngrok-serve task:



Using the free tool [Postman](https://www.postman.com/), create a new request to the point endpoint:

* set the Content-Type header to application/json on the **Headers** tab:



* add the following JSON to the **Body** tab and select the **raw** option (*make sure to update the URL of the image in the sections[0].activityTile property so the image renders when the message is set to Microsoft Teams*):

{

"@type": "MessageCard",

"@context": "http://schema.org/extensions",

"themeColor": "0076D7",

"summary": "Larry Bryant created a new task",

"sections": [

{

"activityTitle": "![TestImage](https://aaaaedee.ngrok.io/Content/Images/default.png)Larry Bryant created a new task",

"activitySubtitle": "On Project Tango",

"activityImage": "https://myfirstteamsconnector.azurewebsites.net/static/img/image5.png",

"facts": [

{

"name": "Assigned to",

"value": "Unassigned"

},

{

"name": "Due date",

"value": "Mon May 01 2017 17:07:18 GMT-0700 (Pacific Daylight Time)"

},

{

"name": "Status",

"value": "Not started"

}

],

"markdown": true

}

],

"potentialAction": [

{

"@type": "ActionCard",

"name": "Add a comment",

"inputs": [

{

"@type": "TextInput",

"id": "comment",

"isMultiline": false,

"title": "Add a comment here for this task"

}

],

"actions": [

{

"@type": "HttpPOST",

"name": "Add comment",

"target": "http://..."

}

]

},

{

"@type": "ActionCard",

"name": "Set due date",

"inputs": [

{

"@type": "DateInput",

"id": "dueDate",

"title": "Enter a due date for this task"

}

],

"actions": [

{

"@type": "HttpPOST",

"name": "Save",

"target": "http://..."

}

]

},

{

"@type": "ActionCard",

"name": "Change status",

"inputs": [

{

"@type": "MultichoiceInput",

"id": "list",

"title": "Select a status",

"isMultiSelect": "false",

"choices": [

{

"display": "In Progress",

"value": "1"

},

{

"display": "Active",

"value": "2"

},

{

"display": "Closed",

"value": "3"

}

]

}

],

"actions": [

{

"@type": "HttpPOST",

"name": "Save",

"target": "http://..."

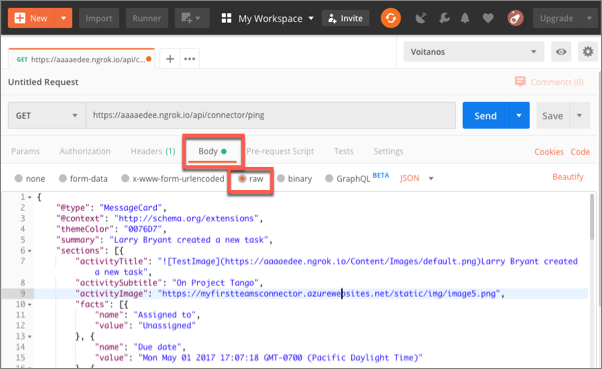
}

]

}

]

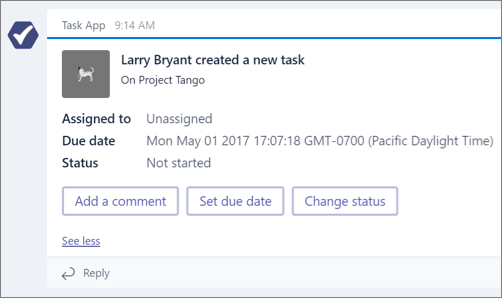
}



**Important**

Connectors, like incoming webhooks, only support Office 365 Connector Cards for messages sent to Microsoft Teams. Adaptive cards aren't supported when sending messages with cards when using Connectors or incoming webhooks.

Select the **Send** button in Postman. When you go back to the channel, you will see the card displayed as a message in the team:



## Summary

In this unit, you learned how to create an Office 365 Connector and add it to Microsoft Teams.