Microsoft 365

팀즈 개발자양성 과정

3-3. Webhooks & O365 Connector

Kwang Jin Jung

Customer Engineer

10/12/2021

Contents

Learning objectives

- Demonstrate how to create and use an outgoing webhook for a Microsoft Teams channel
- Demonstrate how to create and use an incoming webhook for a Microsoft Teams channel
- Demonstrate how to create, register, and use an Office 365 Connector for Microsoft Teams

01. Webhooks & Connectors in Microsoft Teams


Microsoft Teams apps
Tabs (embedded web content)
Authenticate with third parties
Display <iframe> content
Display task modules
Display SharePoint pages and web parts
Deep linking
Request device permissions
Display Power Apps
Display Power BI data
Bots (conversational interface)
Delete messages
Update messages
Send files to users
Send cards
Display task modules
Send proactive messages
Get user, group chat, and team details
Calling and meeting
Authenticate with third parties
Graph (REST API for using Teams data in your app)
Messaging extensions (compose box add-ons)
Send cards
Unfurl links
Authenticate with third parties
Search and display external data
Take action on messages
Display task modules
Webhooks & connectors (external notifications and messages)
Office 365 Connector (incoming webhook)
Customize user experience
Send cards
Refresh cards
Collect user input
Incoming webhooks
Send cards
Collect user input
Refresh cards
Outgoing webhooks
Post data to external app
Request data from external app
Send cards

Entry points for Teams apps

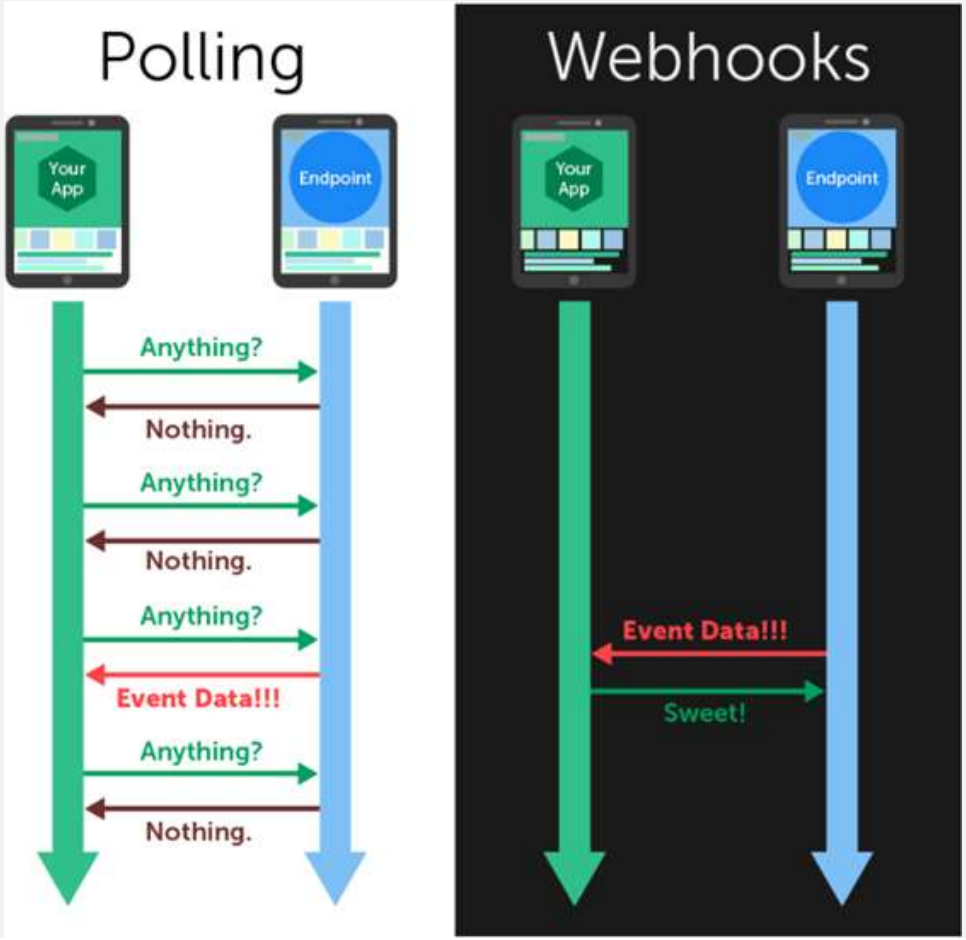
- **Shared app experiences**：Team, channel, and chat are collaboration spaces. Apps in these contexts are available to everyone in that space. Collaboration spaces typically focus on additional workflows or unlocking new social interactions.
  - **Tabs** provide a full-screen embedded web experience configured for the team, channel, or group chat. All members interact with the same web-based content, so a stateless single-page app experience is typical.
  - **Messaging extensions** are shortcuts for inserting external content into a conversation or taking action on messages without leaving Teams. Link unfurling provides rich content when sharing content from a common URL.
  - **Bots** interact with members of the conversation through chat and responding to events, such as adding a new member or renaming a channel.
  - **Webhooks and connectors** allow an external service to post messages into a conversation and users to send messages to a service.
  - **Microsoft Graph REST API** for getting data about teams, channels, and group chats to help automate and manage Teams processes.

- **Personal app experiences**：focus on interactions with a single user. The experience in this context is unique to each user.
  - **Bots** have one-on-one conversations with a user. Bots that require multi-turn conversations or provide notifications relevant only to a specific user are best suited in personal apps.
  - **Tabs** provide a full-screen embedded web experience that is meaningful to the user looking at it.

About Webhook

*A **webhook** in web development is a method of augmenting or altering the behavior of a web page or web application with custom callbacks. The format is usually JSON. The request is done as an HTTP POST request. (Wikipedia)*

- Event missing by App Failed
- Event duplicate by App not send response
- Too many event



Secure Webhook

Security Checklist

Threat	Solution
Payload Exposure	HTTPS webhook URLs with SSL Encryption
Attack from unknown Webhook source	Authentication taken and whitelisting of webhook source IPs
Webhook interception and redirection to unknown destinations	Client verification using TLS
Webhook payload corruption	Message verification using HMAC signatures
Replay attacks	Timestamped messages

Webhooks & Connectors in Microsoft Teams

- Webhooks and connectors help to connect the web services to channels and teams in Microsoft Teams.
- Webhooks are user defined HTTP callback that notifies users about any action that has taken place in the Microsoft Teams channel.
- It is a way for an app to get real time data.
- Connectors allow users to subscribe to receive notifications and messages from your web services.
- They expose an HTTPS endpoint for your service to post messages in the form of cards.

Outgoing Webhooks

- Webhooks help Teams to integrate with external apps.
- With Outgoing Webhooks, you can send text messages from a channel to the web services.
- After configuring the Outgoing Webhooks, users can @mention Outgoing Webhook and send a message to web services.
- The service responds within ten seconds to the message with a text or a card.
- Outgoing Webhooks are configured on a per team basis and cannot be included as part of a normal Teams app.

Connectors

- Connectors allow users to subscribe to receive notifications and messages from the web services. They expose the HTTPS endpoint for the service to post messages to Teams channels, typically in the form of cards.

Incoming Webhooks

Incoming Webhooks help in posting messages from apps to Teams. If Incoming Webhooks are enabled for a team in any channel, it exposes the HTTPS endpoint, which accepts correctly formatted JSON and inserts the messages into that channel. For example, you can create an Incoming Webhook in your DevOps channel, configure your build, and simultaneously deploy and monitor services to send alerts.

Office 365 Connectors

Office 365 Connectors allow you to create a custom configuration page for your Incoming Webhook and package them as part of a Teams app. You send messages primarily using Office 365 Connector cards and have the ability to add a limited set of card actions to them. For example, a weather connector that allows users to select a location and a time of day, to receive updates about tomorrow's weather. They are configured on a channel level but are installed at a team level.

Rate limiting for connectors

- Application rate limits control the traffic that a connector or an Incoming Webhook is permitted to generate on a channel.
- These limits are in place to reduce spamming a channel by a connector and ensures an optimal experience to users.

[Transactions per seconds thresholds]

Time in seconds	Maximum allowed requests
1	4
30	60
3600	100
7200	150
86400	1800

02. Outgoing Webhooks

Outgoing webhooks & Microsoft Teams

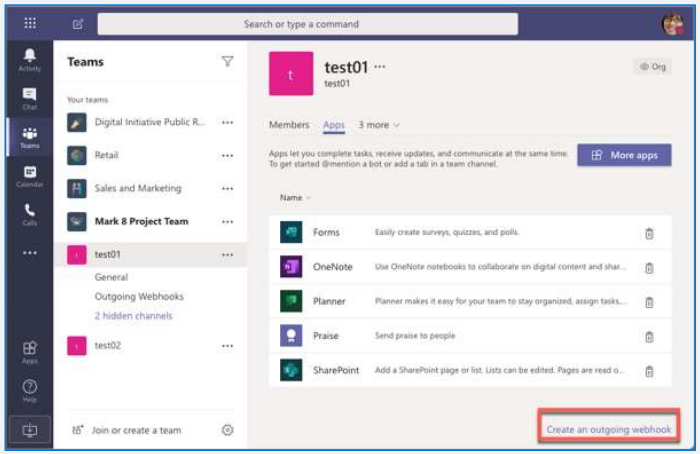
Conceptually work the same way as bots except you don't have to register them with the Microsoft Bot Framework

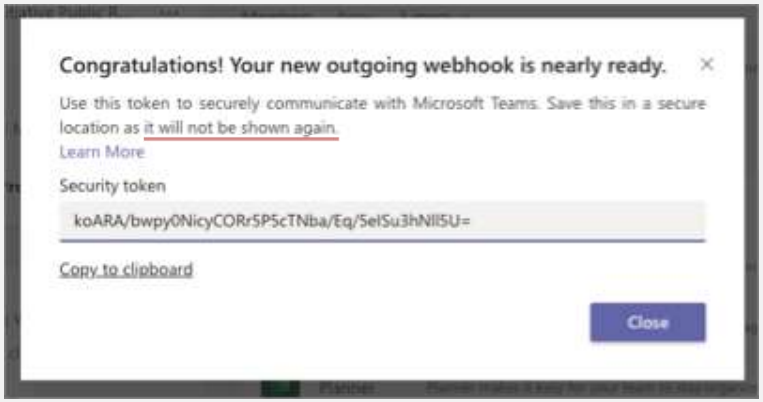
Key features:

- **Scoped configuration**: scoped on a team-by-team basis
- **Reactive messaging**: must be @mentioned
- **Standard HTTP message exchange**: responses appear in same reply chain as original message
- **Limited Microsoft Teams API message support**: messages sent to web service, but don't have access to APIs like list of channels or the channel roster in a team

Add outgoing webhooks to Microsoft Teams

- Create web service that accepts HTTP POST requests with a JSON payload
- Ensure web service validates the message using the provided security token & HMAC auth string
- Register outgoing webhook with Microsoft Teams team

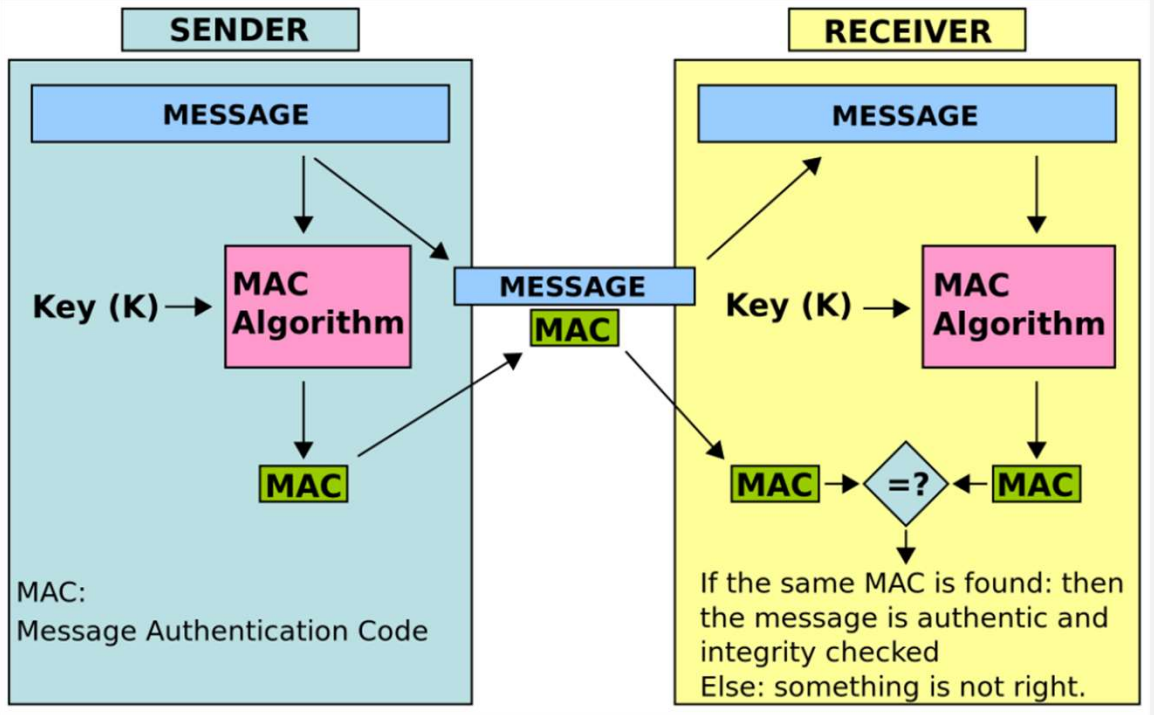




Add outgoing webhooks to Microsoft Teams

HMAC (keyed-hash message authentication code, RFC2104)

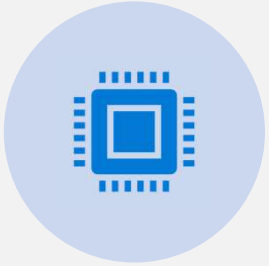
HMAC does not encrypt the message. Instead, the message (encrypted or not) must be sent alongside the HMAC hash. Parties with the secret key will hash the message again themselves, and if it is authentic, the received and computed hashes will match.



Authenticating messages from Microsoft Teams

After registering an outgoing webhook, Microsoft Teams displays a security token

Each message sent to your web service includes an HMAC in the HTTP request authorization header

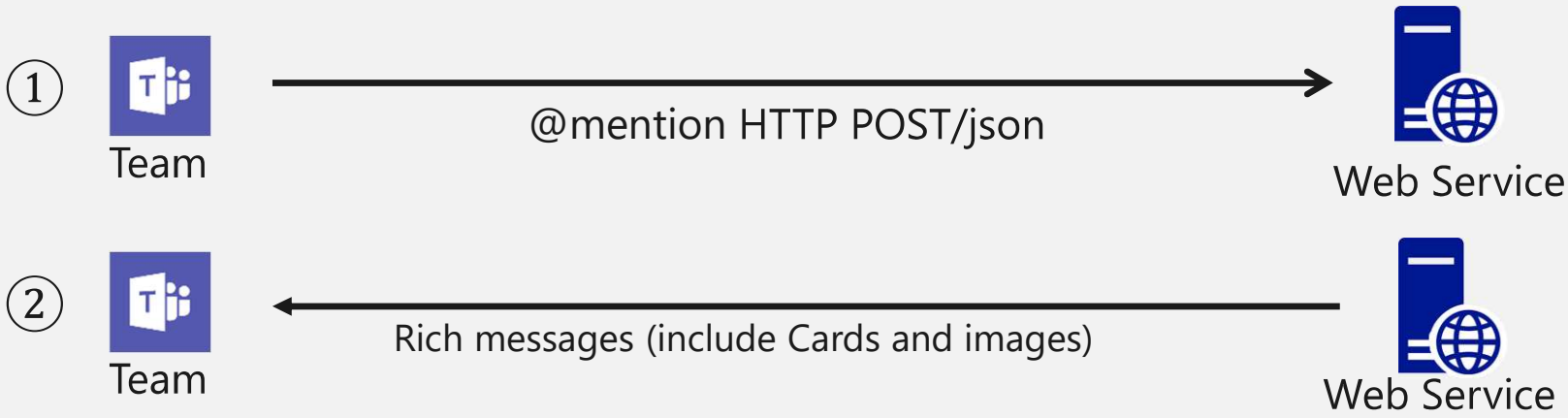


Use the security token to create a HMAC in your web service

Verify your HMAC code matches the one include in the message received from Microsoft Teams

DEMO

Create outgoing webhooks



[LAB scenario]

1. Create Web Service with Node.js
2. Create Outgoing Webhook in Team.
3. User mention @Planet Details <*Name*> in Channel of Team.
4. Web Service return Planet information with defined data.

https://docs.microsoft.com/en-us/learn/modules/msteams-webhooks-connectors/3-exercise-outgoing-webhooks

Prerequisites

- Node.js - v12.* (or higher)
- NPM (installed with Node.js) - v6.* (or higher)
- Gulp - v4.* (or higher)
- Yeoman - v3.* (or higher)
- Yeoman Generator for Microsoft Teams - v3.2.0 (or higher)
- Visual Studio Code

Setup Environment for Yeoman Teams Generator

- Install Node.js
- Install yeoman and gulp cli

  npm install yo gulp-cli --global
- Install Microsoft Teams app generator

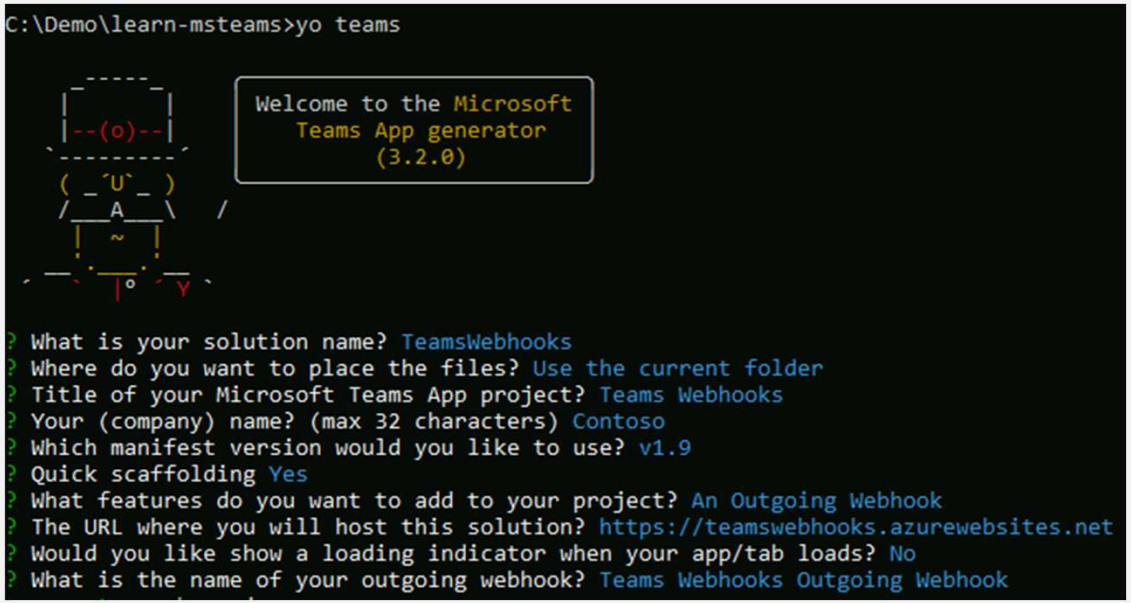
  npm install generator-teams --global

Demo

- Create Teams app

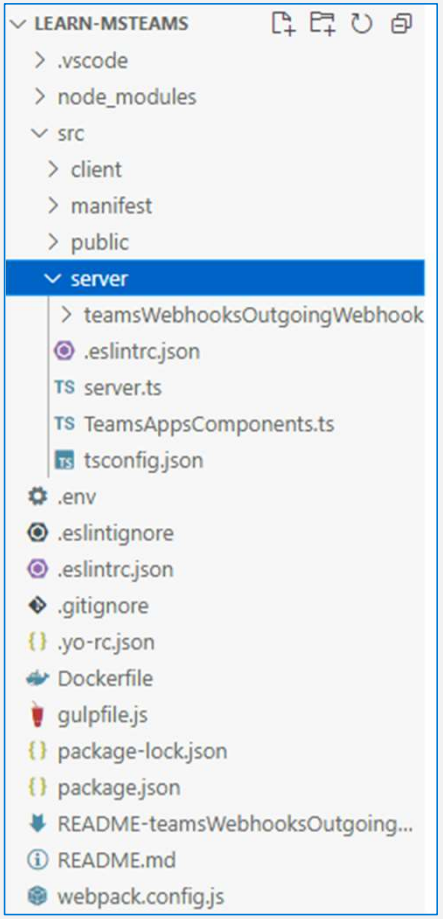
  mkdir <learn-msteams>

  yo teams



- Install additional package : lodash (https://lodash.com)

  npm install lodash -S



Demo

- ./src/server/teamsWebhooksOutgoingWebhook/TeamsWebhooksOutgoingWebhook.ts

```
const msgHash = "HMAC " + crypto
    .createHmac("sha256", Buffer.from(securityToken as string, "base64"))
    .update(msgBuf)
    .digest("base64");

if (msgHash === auth) {
    // Message was ok and verified
    message.text = `Echo ${incoming.text}`;
} else {
    // Message could not be verified
    message.text = "Error: message sender cannot be verified";
}
```

- ./.env

```
# Security token for the default outgoing webhook
SECURITY_TOKEN=
```

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

Demo

- Create new file **planets.json**, **planetDisplayCard.json** in ./src/server/teamsWebhooksOutgoingWebhook/

  planets.json : Data json

  planetDisplayCard.json : Adaptive Cards template

- Add ***import*** statement in

  ./src/server/teamsWebhooksOutgoingWebhook/TeamsWebhooksOutgoingWebhook.ts

```
import * as builder from "botbuilder";
import * as express from "express";
import * as crypto from "crypto";
import { OutgoingWebhookDeclaration, IOutgoingWebhook } from "express-msteams-host";
import { find, sortBy } from "lodash";
```

Demo

- Update requestHandler() method in *TeamsWebhooksOutgoingWebhook* class
  ./src/server/teamsWebhooksOutgoingWebhook/TeamsWebhooksOutgoingWebhook.ts

```
// create the response, any Teams compatible responses can be used
const message: Partial<builder.Activity> = {
  type: builder.ActivityTypes.Message
};
```

```
// create the response, any Teams compatible responses can be used
let message: Partial<builder.Activity> = {
  type: builder.ActivityTypes.Message
};
```

```
if (msgHash === auth) {
  // Message was ok and verified
  message.text = `Echo ${incoming.text}`;
} else {
  // Message could not be verified
  message.text = "Error: message sender cannot be verified";
}
```

```
if (msgHash === auth) {
  // Message was ok and verified
  const scrubbedText = TeamsWebhooksOutgoingWebhook.scrubMessage(incoming.text);
  message = TeamsWebhooksOutgoingWebhook.processAuthenticatedRequest(scrubbedText);
} else {
  // Message could not be verified
  message.text = "Error: message sender cannot be verified";
}
```

Demo

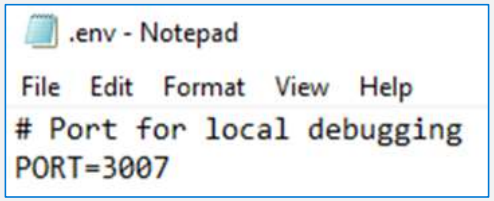
- Add method ***getPlanetDetailCard(), processAuthenticatedRequest(), scrubMessage()*** in *TeamsWebhooksOutgoingWebhook* class

```
import * as builder from "botbuilder";
import * as express from "express";
import * as crypto from "crypto";
import { OutgoingWebhookDeclaration, IOutgoingWebhook } from "express-msteams-host";
import { find, sortBy } from "lodash";

/**
 * Implementation for Teams Webhooks Outgoing Webhook
 */
@OutgoingWebhookDeclaration("/api/webhook")
export class TeamsWebhooksOutgoingWebhook implements IOutgoingWebhook {

  /**
   * Implement your outgoing webhook logic here
   * @param req the Request
   * @param res the Response
   * @param next
   */
  public requestHandler(req: express.Request, res: express.Response, next: express.NextFunction) ...

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

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

  private static scrubMessage(incomingText: string): string ...
}
```

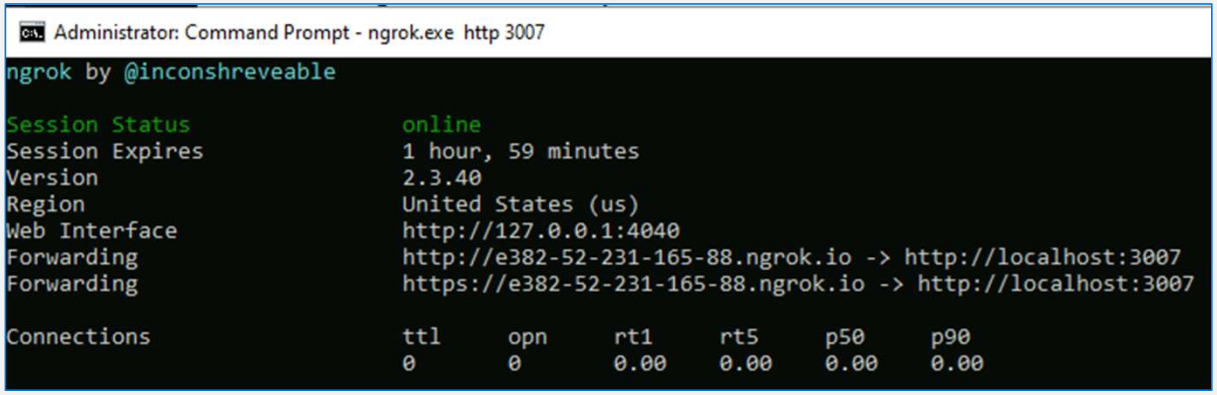
Demo

- Test outgoing webhook

  gulp serve : Execute server



  ngrok http 3007 : HTTP tunnel

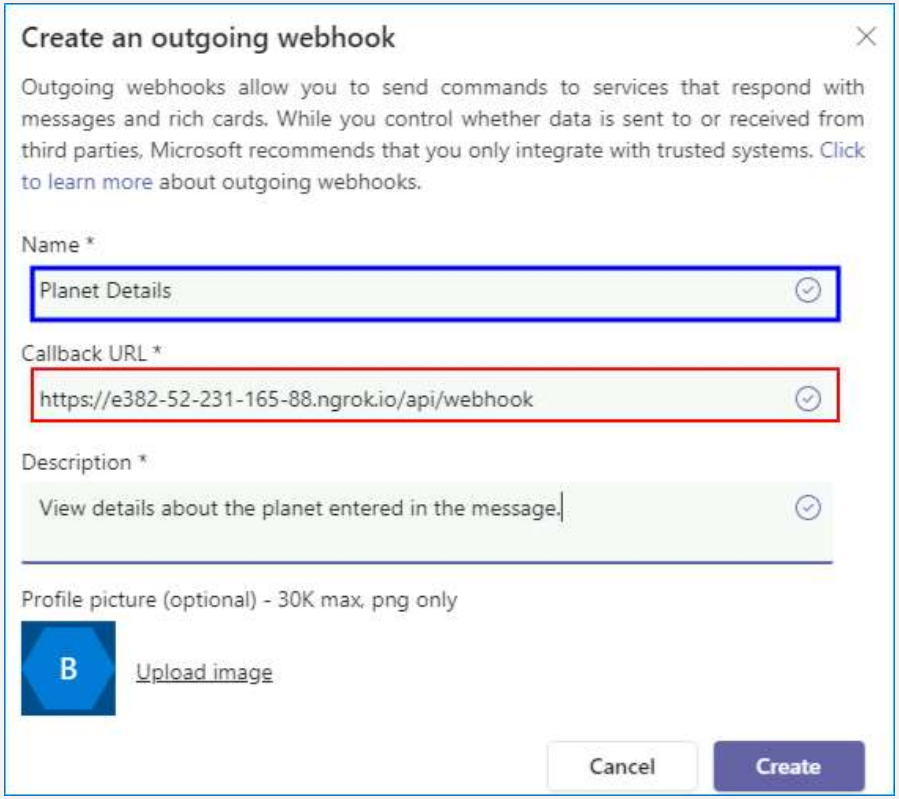


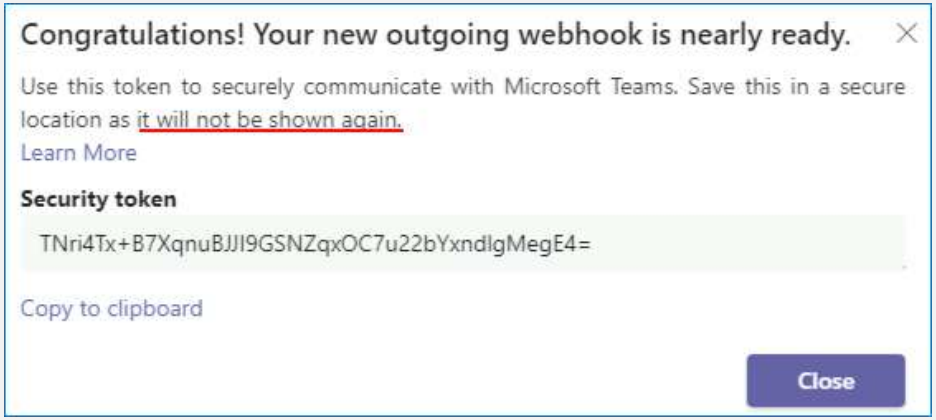
  Locate and execute ngrok.exe in *<learn-msteams>* folder

Demo

- Create outgoing Webhook
- Click “Create an outgoing webhook” on Apps page in Manage Team

```
Forwarding https://e382-52-231-165-88.ngrok.io -> http://localhost:3007
```

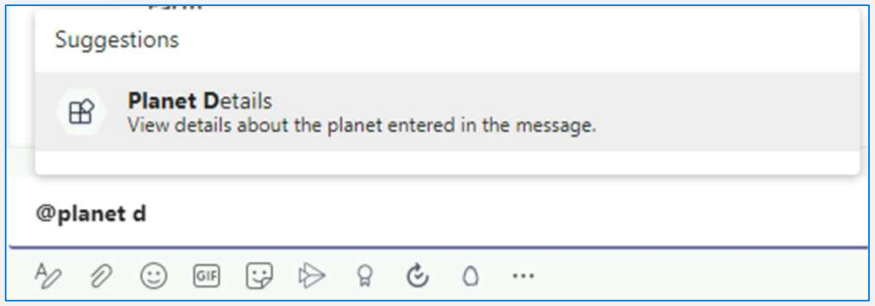


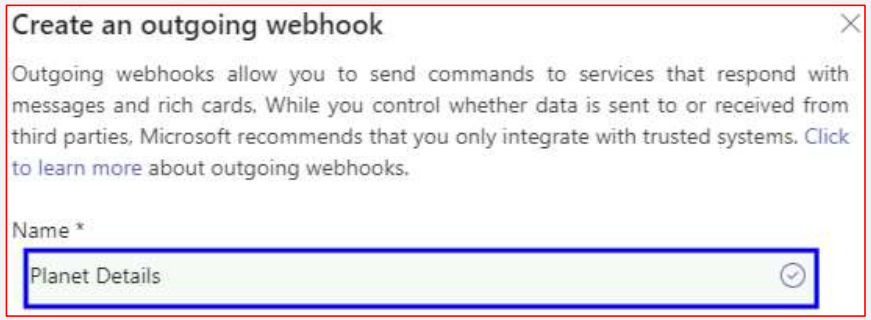


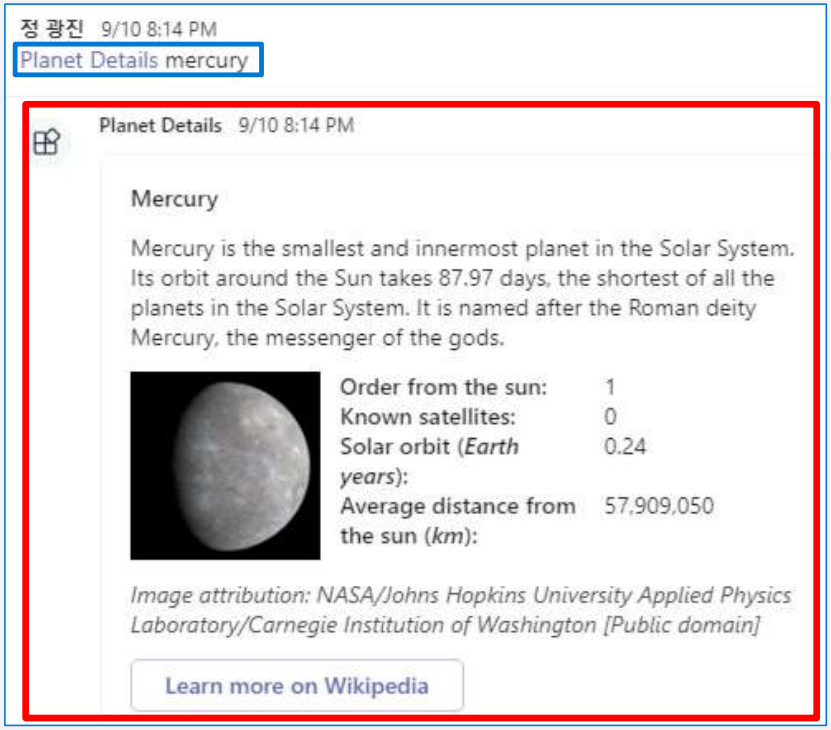
Restart gulp serve

Demo

- Test in Teams







03. Incoming Webhooks

Incoming webhooks & Microsoft Teams

Incoming webhooks are a special type of Connector

Provide simple way for external apps to share content & team channels

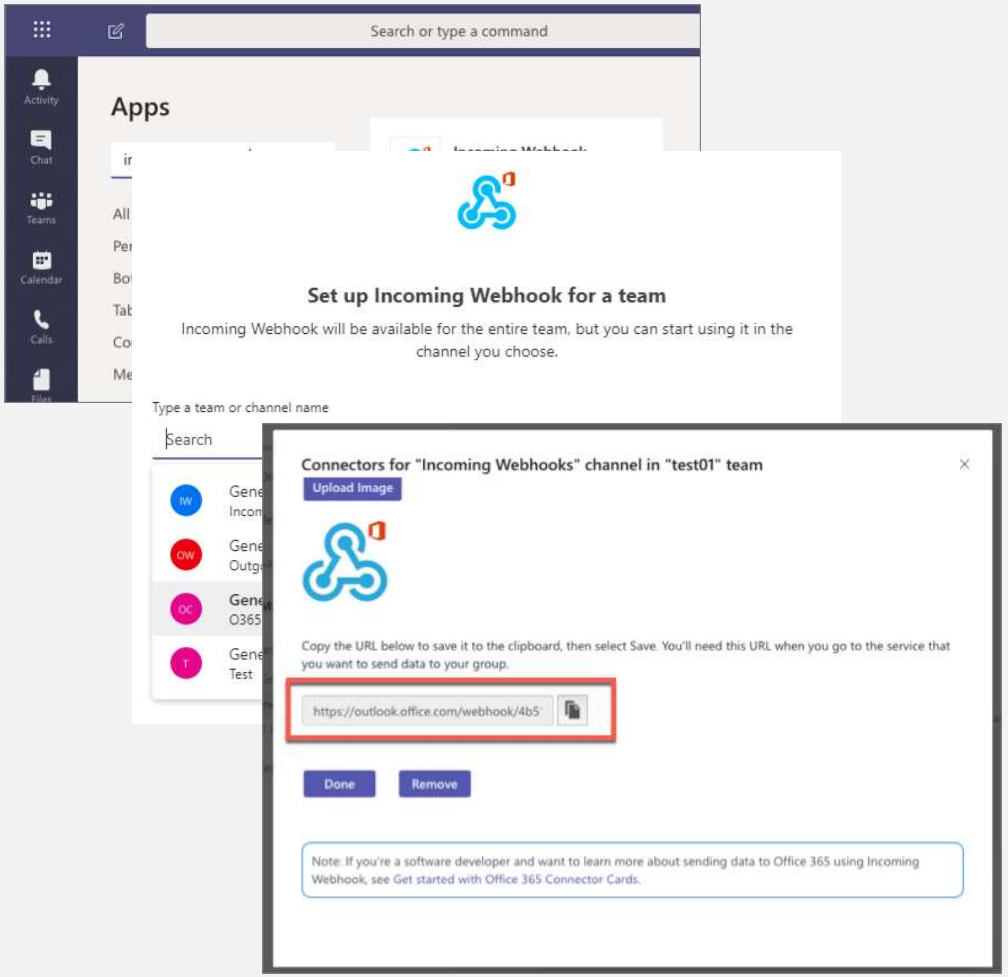
Microsoft Teams provide unique URL for a channel that your web service can post to

Key features:

- **Scoped configuration**: scoped & configured at the channel level
- **Secure resource definition**: messages included as JSON payloads;
  prevents malicious code injection
- **Actionable messaging support**: send messages as text-based messages or rich cards
- **Independent HTTPs messaging support**: present clear & consistent messages with cards
- **Markdown support**: support for rich formatting of text

Add incoming webhooks to Microsoft Teams

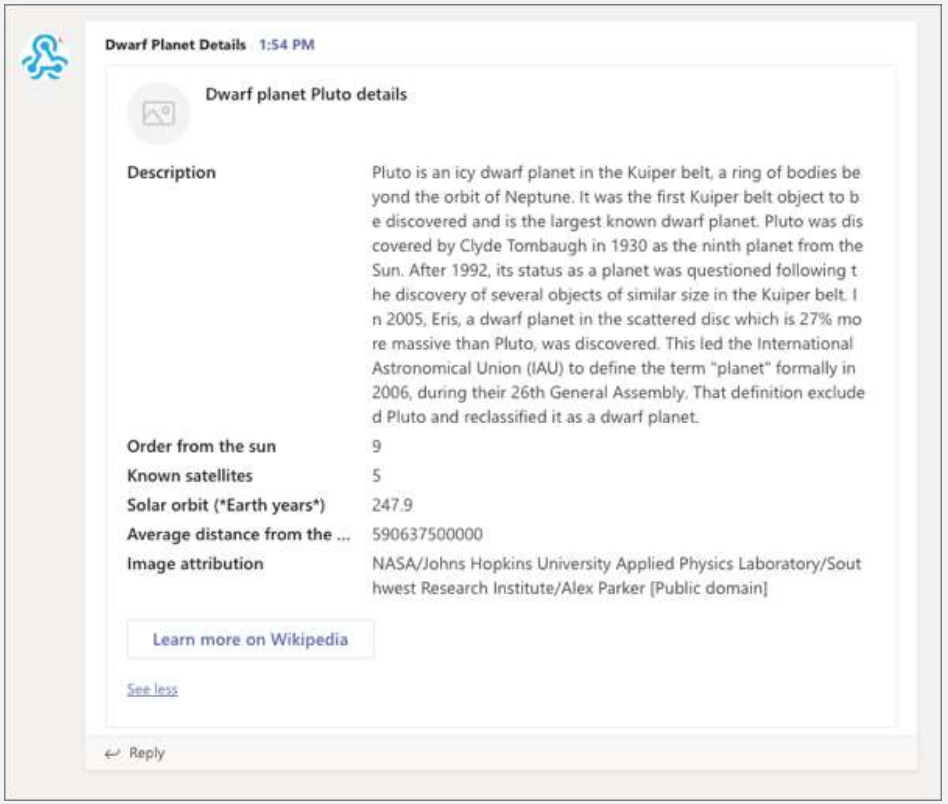
- Create web service that can send HTTP POST requests to an HTTPS endpoint
- Add the **Incoming webhook** app to a channel & configure it – copy the provided endpoint



Add incoming webhooks to Microsoft Teams

- Your web service sends HTTP POST request with a JSON payload to the provided endpoint
- Messages are added to the **Conversations** tab in the configured channel

* Need Office 365 Connector Cards
  (Not support adaptive card's Action.submit)



Add incoming webhooks to Microsoft Teams

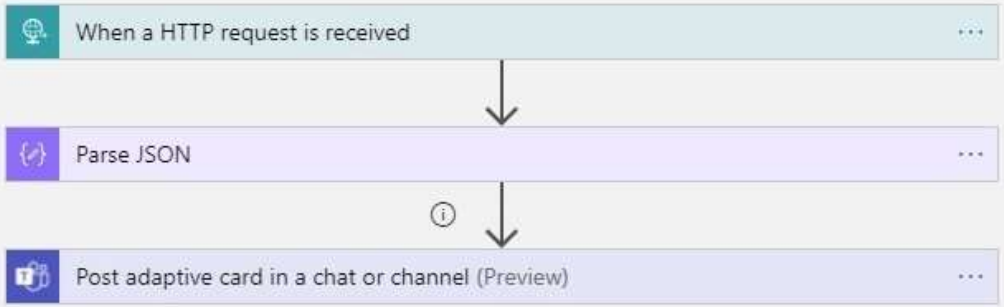
Adaptive, hero, list, Office 365 Connector, receipt, signin, and thumbnail cards and card collections are supported in bots for Microsoft Teams. They are based on cards defined by the Bot Framework, but Teams does not support all Bot Framework cards and has added some of its own.

[Types of cards - Teams | Microsoft Docs]

Card type	Bots	Message extension previews	Message extension results	Task modules	Outgoing Webhooks	Incoming Webhooks	Office 365 Connectors
Adaptive Card	✓	×	✓	✓	✓	✓	×
Office 365 Connector card	✓	×	✓	×	✓	✓	✓
Hero card	✓	✓	✓	×	✓	✓	×
Thumbnail card	✓	✓	✓	×	✓	✓	×
List card	✓	×	×	×	✓	✓	×
Receipt card	✓	×	×	×	×	✓	×
Signin card	✓	×	×	×	×	×	×

Incoming Webhooks Security

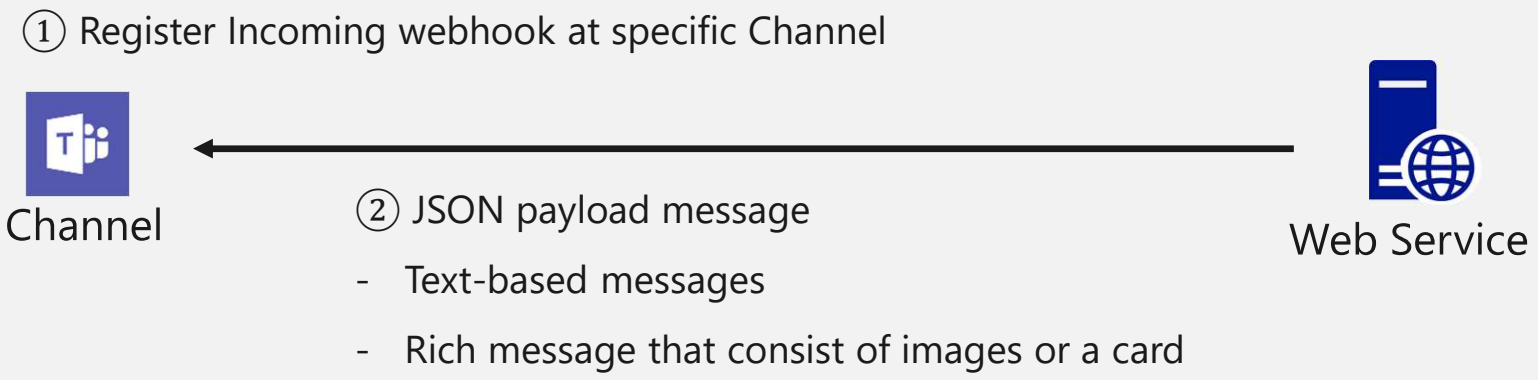
- Incoming webhooks are a simple integration solution but :
  - Better control over who can create and use a webhook
  - More options to authenticate the event emitter
  - Protect users from the content published into Teams
- Azure Logic App



https://www.linkedin.com/pulse/bring-microsoft-teams-incoming-webhook-security-next-level-kinzelin

DEMO

Create incoming webhooks


① Register Incoming webhook at specific Channel
Channel
② JSON payload message
- Text-based messages
- Rich message that consist of images or a card
Web Service

Prerequisites

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.

[Free tools]

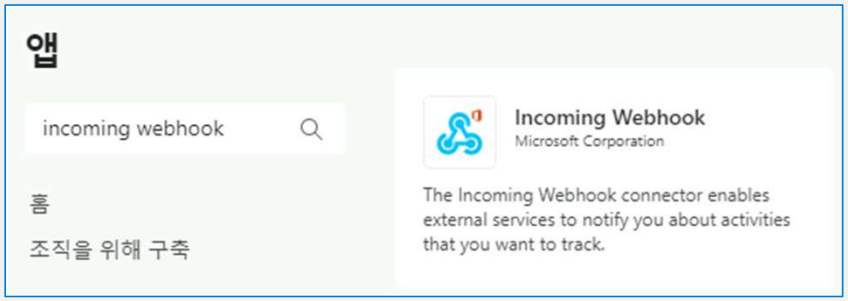
- cURL : Interactive cli tool for HTTP inspection
- Wuzz : Interactive cli tool for HTTP inspection
- Postman : Interactive gui tool for HTTP inspection
- Fiddler : Composer menu
- Powershell : Invoke-RestMethod

Demo

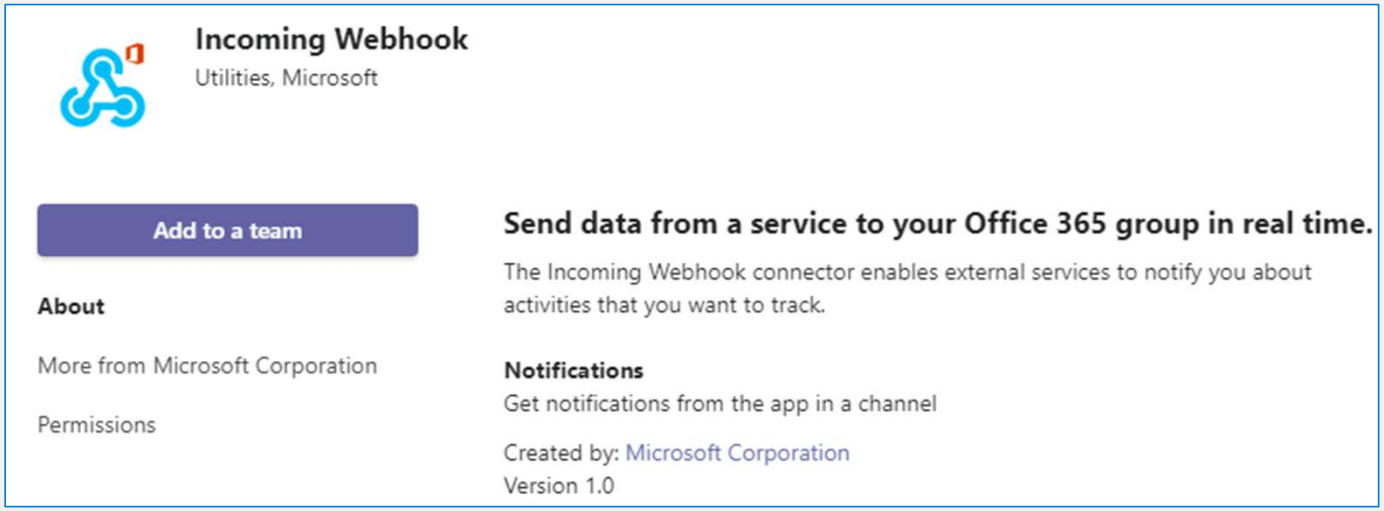
From Channel page, select '+'.

[Add a tab] – Manage apps – More apps

search "incoming webhook"



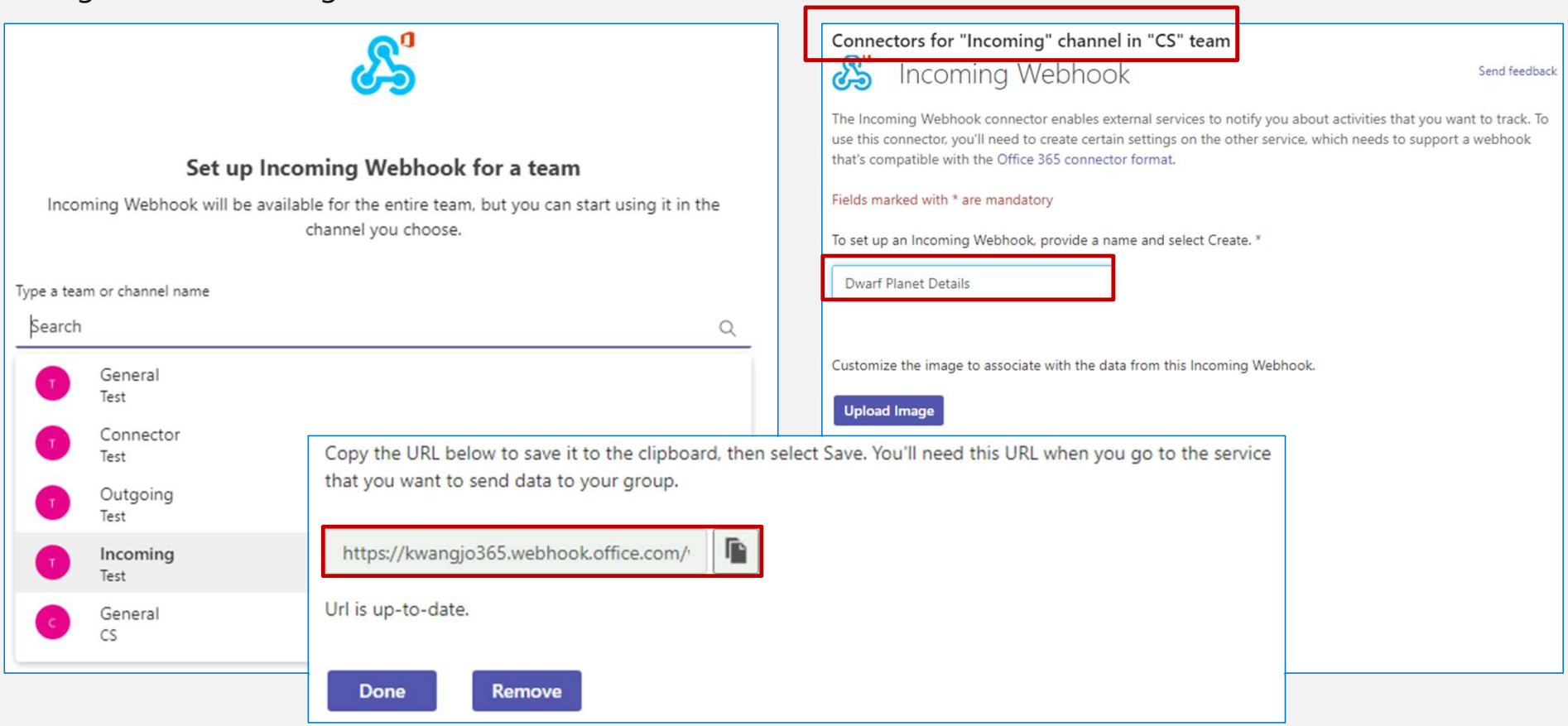
Add to a team



Demo

Select or type channel name for add incoming webhook

Configuration Incoming webhook

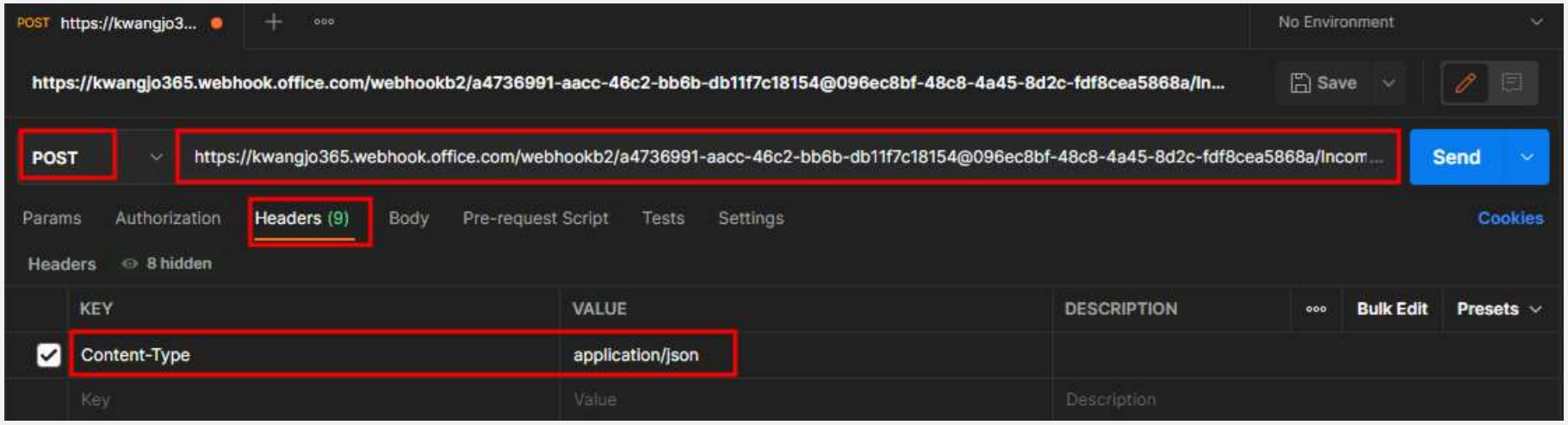


Demo

- Test incoming webhook

  Using “Postman”, create a new request to the point endpoint

  - Set the request to a **POST**
  - Set the endpoint to the webhook endpoint
  - Set the **content-type** header to **application/json** on **Headers** tab



Demo

- Test incoming webhook

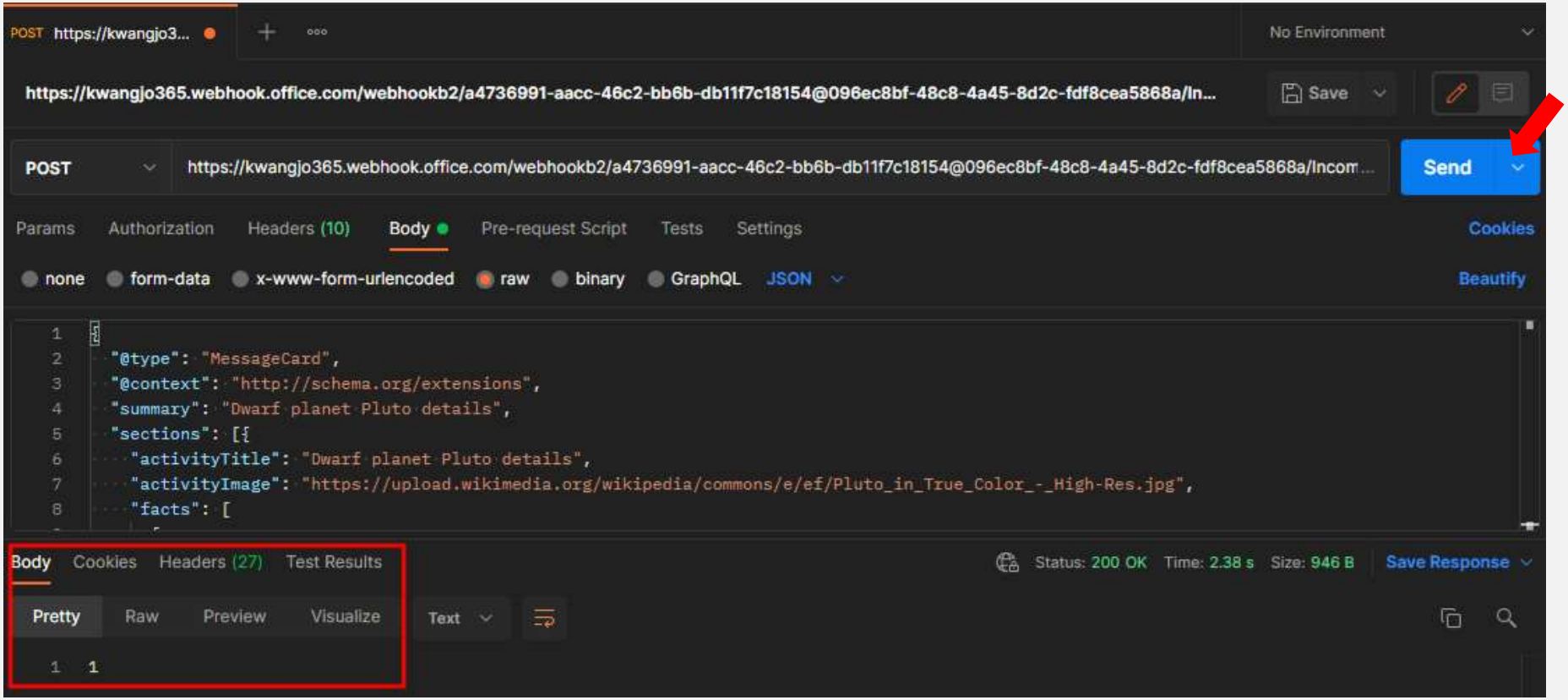
  Using “Postman”, create a new request to the point endpoint

  - Set the **raw** on **Body** tab
  - Add JSON data

Demo

- Test incoming webhook

  Using “Postman”, Check Response’s value is “**1**”



Demo

- Test incoming webhook

  Using “cURL” :

```
// on Windows
curl.exe -H "Content-Type:application/json" -d "{'text':'Hello World'}" <YOUR WEBHOOK URL>
```

- Test incoming webhook

  Using “Powershell” command :

```
Invoke-RestMethod -Method post -ContentType 'Application/Json' -Body '{"text":"Hello World!"}' -
Uri <YOUR WEBHOOK URL>
```

- If the POST succeeds, see a simple **1** output.

cURL download : https://curl.haxx.se/

04. O365 Connector

Office 365 Connectors

Office 365 Connectors are like incoming webhooks except the install & configuration experience is slightly simpler

Connectors have the same capabilities as incoming webhooks

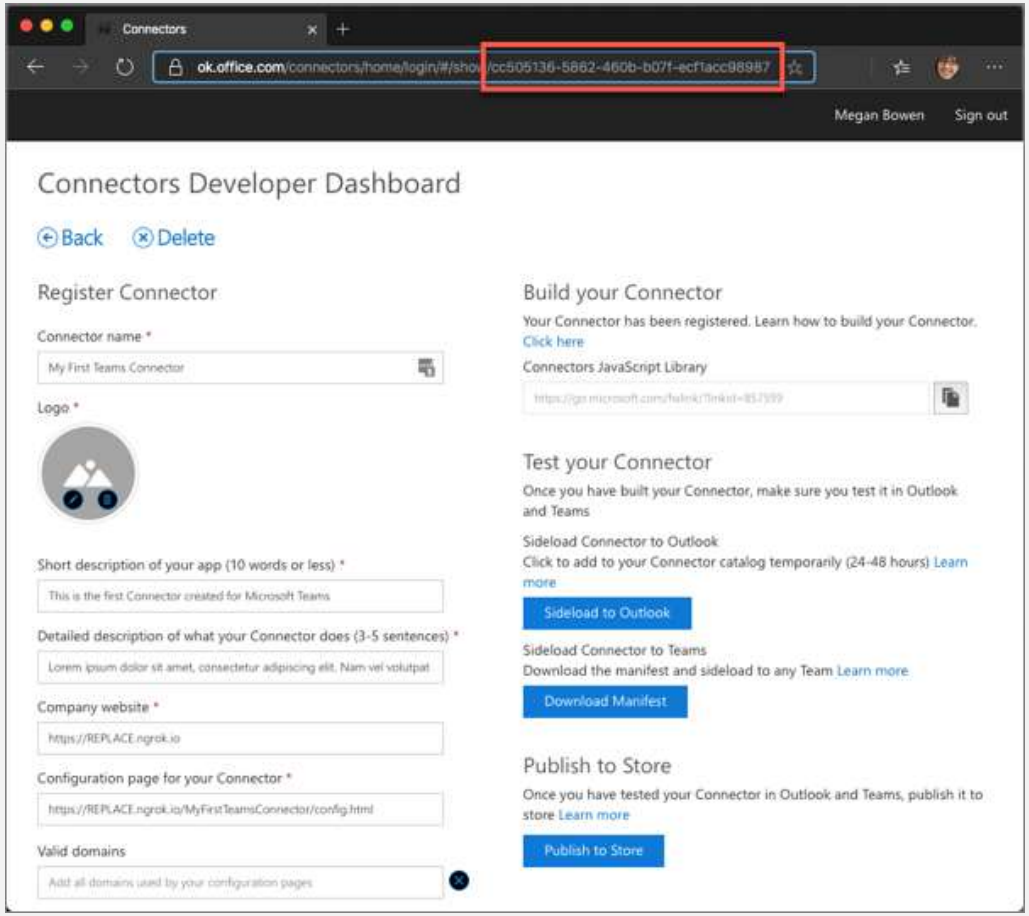
Connectors differ from incoming webhooks in the following ways:

- **Inclusion in AppSource**: once validated by Microsoft, can be included in AppSource for broad distribution
- **Publish to your organization**: publish to your organization's App Store to enable the entire organization to use it
- **Custom configuration experience**: when adding to a channel, custom configuration screen is displayed that enables collection of Connector-specific configuration details

Add an Office 365 Connector to a Microsoft Teams app

- Connectors are distributed to Microsoft Teams in the custom Teams app manifest
- All Connectors must first be registered in the **Connectors Developer Dashboard**
- Once registered, retrieve the ID of the Connector for inclusion in the Teams app manifest
- Go Connectors Developer Dashboard

  https://aka.ms/ConnectorsDashboard



Add an Office 365 Connector to a Microsoft Teams app

- When a Connector is added to a channel, Microsoft Teams displays a configuration page within an IFRAME
- Enables Connector developers to collect Connector-specific configuration data
- When the user selects **Save**, the configuration page submits the request to the web service
- Can include the webhook endpoint Microsoft Teams creates for the Connector

Configuration page specifics

Developers implementing the configuration experience should do the following:

Include the Microsoft Teams JavaScript SDK

- Initialize the SDK by calling `microsoftTeams.initialize()`
- Call `microsoftTeams.settings.setValidityState(true)` to enable Save
- Register `microsoftTeams.settings.registerOnSaveHander()` event handler
- Call `microsoftTeams.settings.setSettings()` to save the connector settings.
- Call `microsoftTeams.settings.getSettings()` to fetch webhook properties.
- Optionally implement the `microsoftTeams.settings.registerOnRemoveHandler()`

Add the Connector to the Microsoft Teams app manifest

```
{
  "manifestVersion": "1.5",
  "id": "5f6df040-493b-11ea-897f-1b87e48d60c9",
  "version": "0.0.1",
  "packageName": "myfirstteamsconnector",
  "connectors": [
    {
      "connectorId": "{{CONNECTOR_ID}}",
      "configurationUrl": "https://{{HOSTNAME}}/myFirstTeamsConnector/config.html",
      "scopes": ["team"]
    }
  ]
}
```

Enable or disable connectors in Teams

- Connect to Exchange Online

  Install-Module -Name ExchangeOnlineManagement
  Import-Module ExchangeOnlineManagement
  Connect-ExchangeOnline -UserPrincipalName [admin@[tenant-id].onmicrosoft.com]

- To disable connectors for the tenant

  Set-OrganizationConfig –ConnectorsEnabled:$false

- To disable actionable messages for the tenant

  Set-OrganizationConfig –ConnectorsActionablesEnabled:$false

- To enable connectors for Teams

  Set-OrganizationConfig –ConnectorsEnabled:$true

  Set-OrganizationConfig –ConnectorsEnabledForTeams:$True

  Set-OrganizationConfig –ConnectorsActionableMessagesEnabled:$true

DEMO

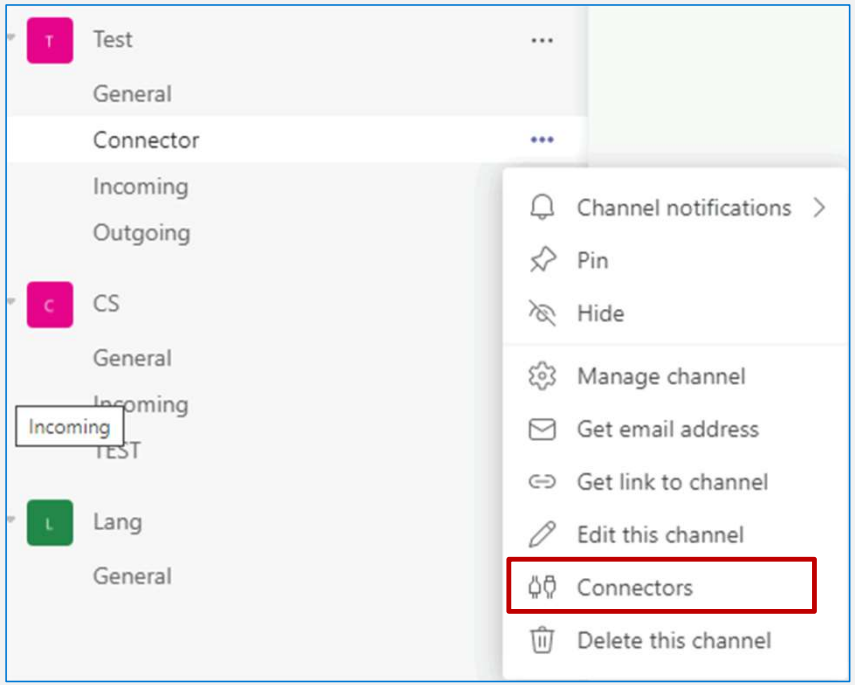
Create and add Office 365 Connectors to teams

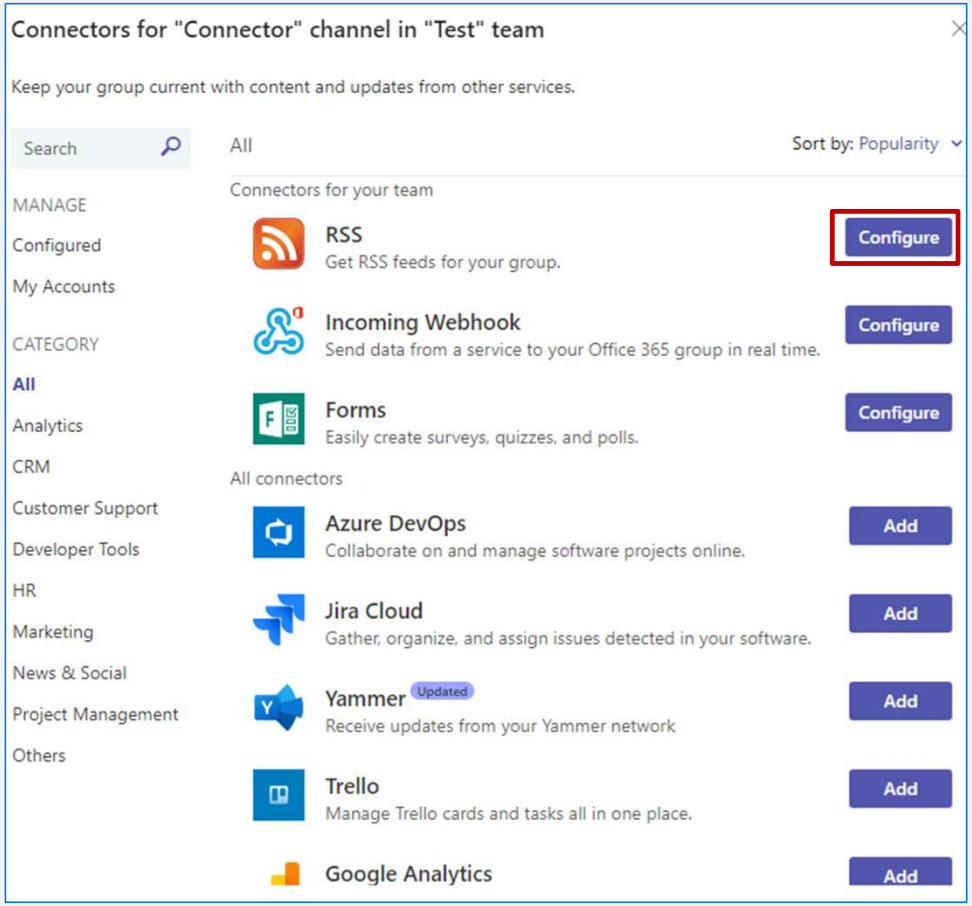
Demo : Add a RSS connector to a channel

1. To add a connector to a channel, click the **ellipses** (**...**), on the right of a channel name, then click **Connectors**.
2. select from a variety of available connectors, and then click **Add**.
3. Fill in the required information of the selected connector and click **Save**. Each connector requires a diverse set of information to function properly, and some may require you to sign in to the service using the links provided on the connector configuration page.
4. Data provided by the connector is automatically posted to the channel.

Demo : Add a RSS connector to a channel

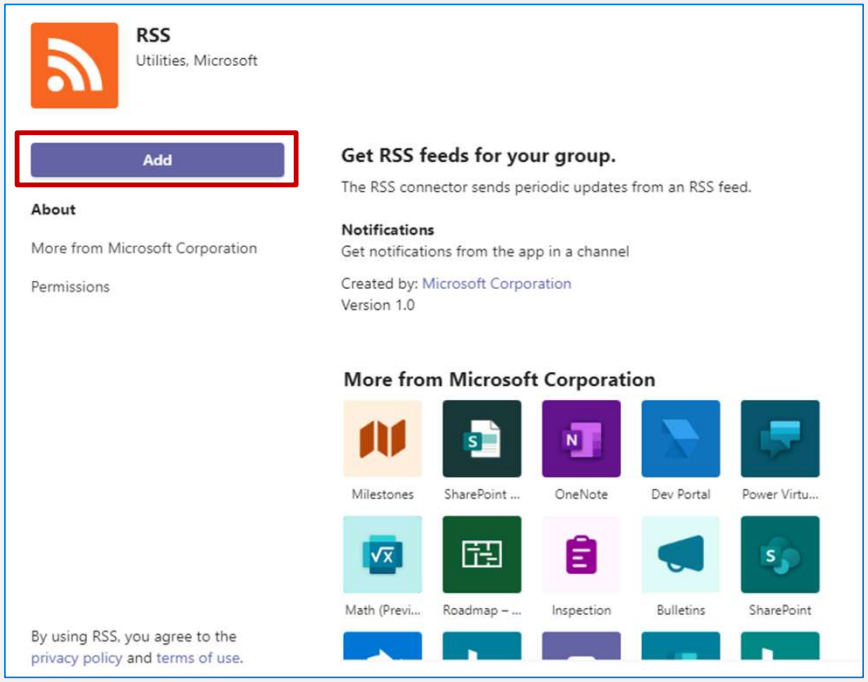
1. To add a connector to a channel, click the **ellipses** (**...**), on the right of a channel name, then click **Connectors.**
2. Select **RSS** and then click **Add**.

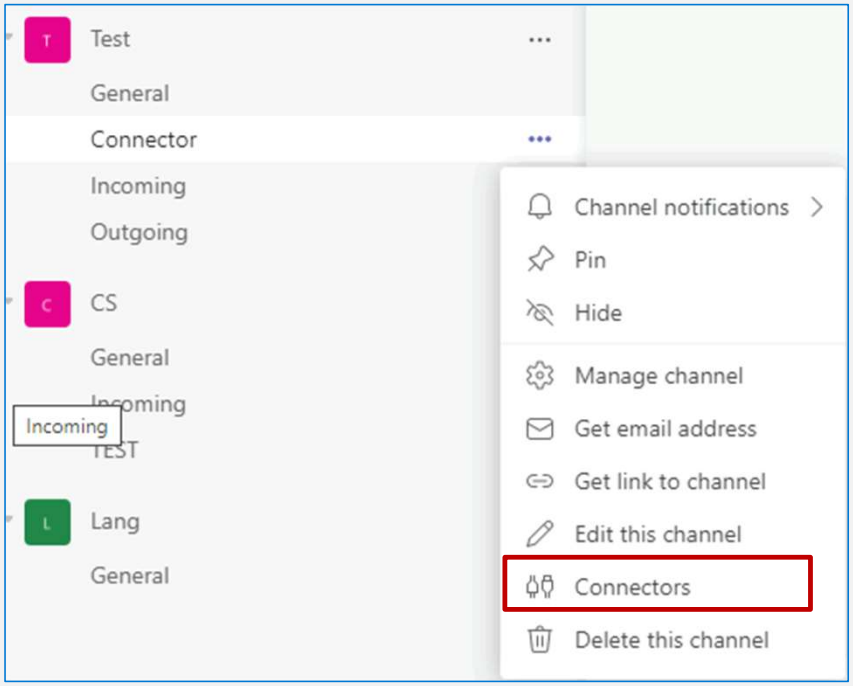




Demo : Add a RSS connector to a channel

3. Click **Add**
4. Re-launch **Connectors**

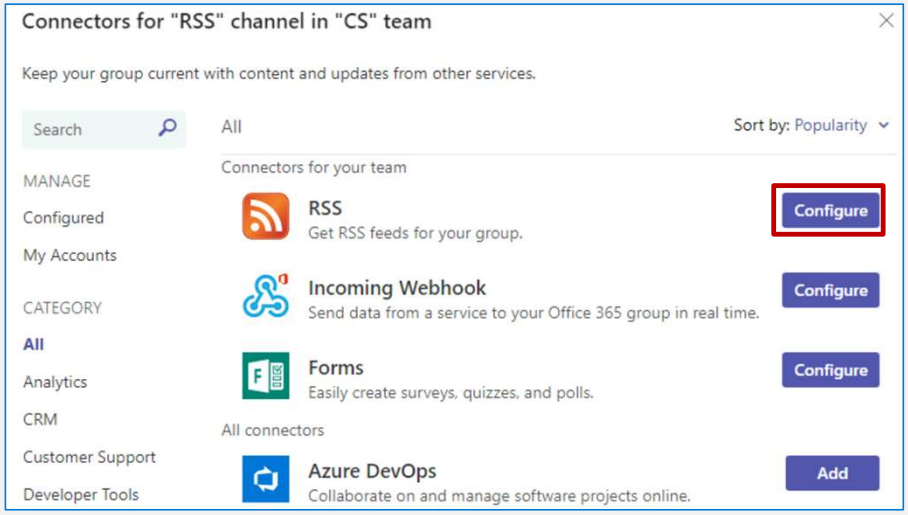


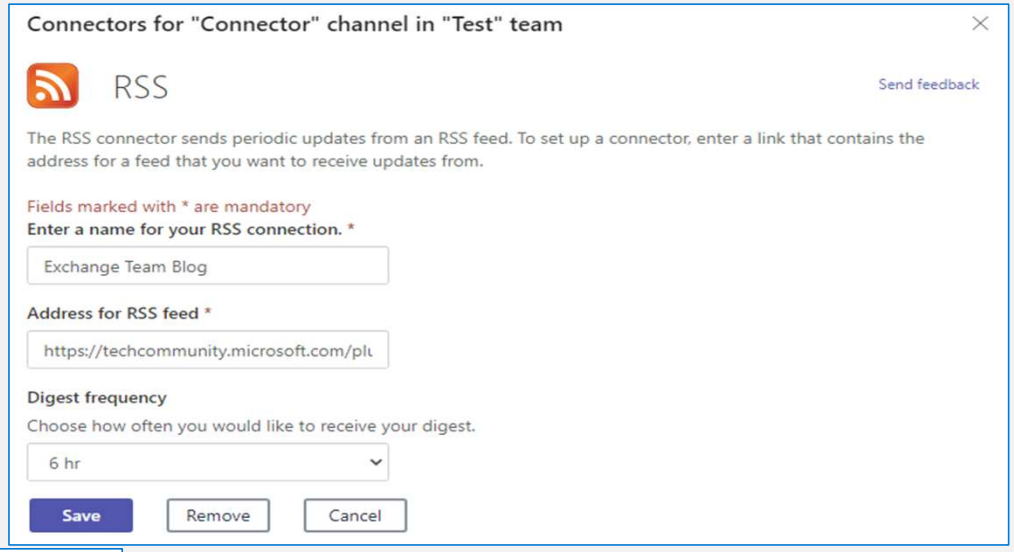


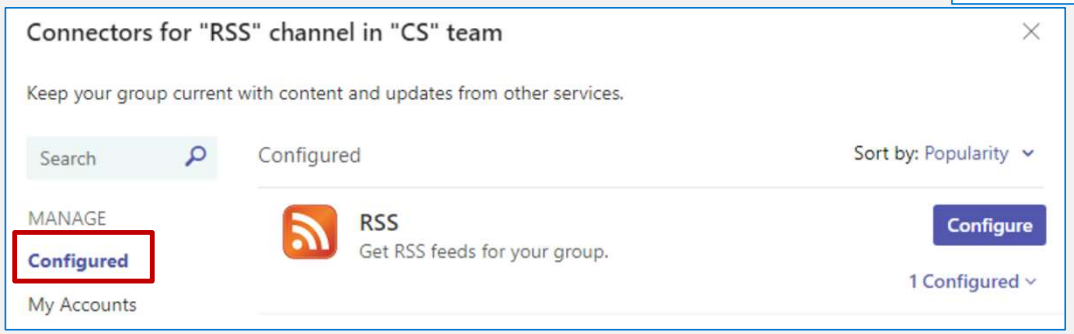
Demo : Add a RSS connector to a channel

5. Click **Configure** for RSS

6. Fill in the required information and click **Save**

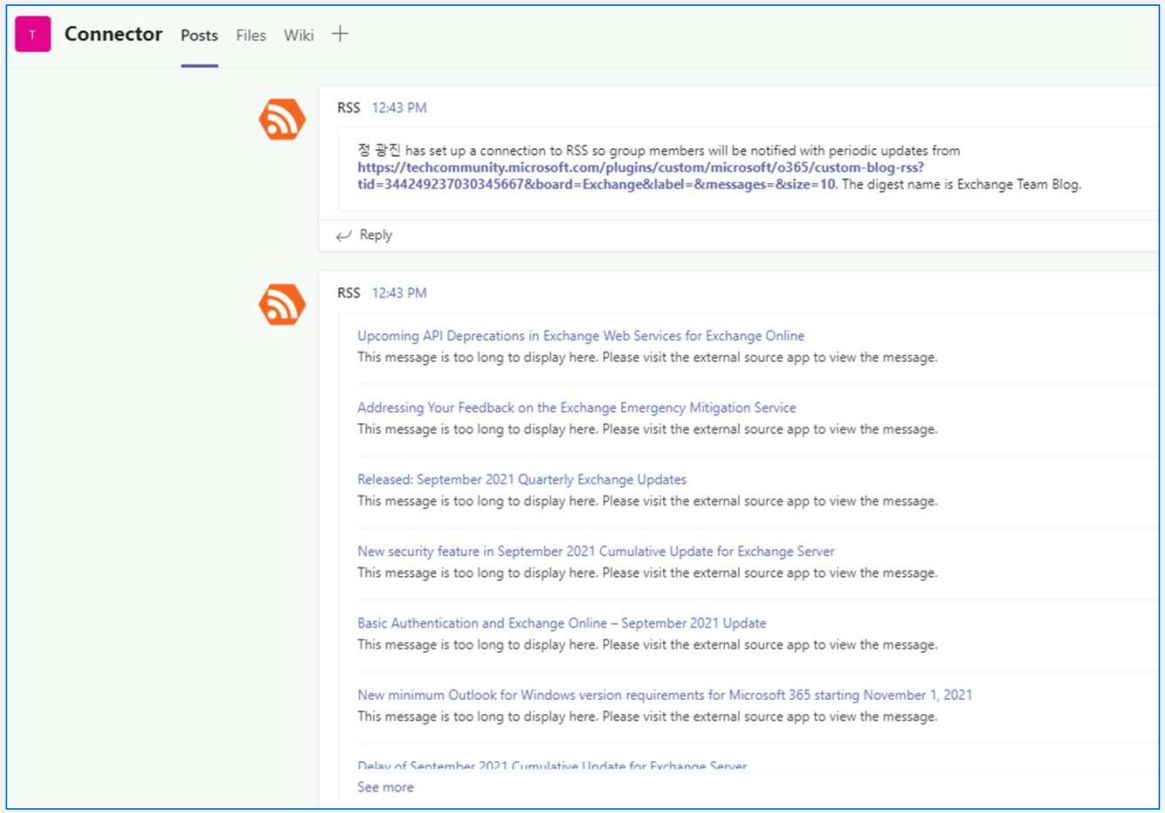




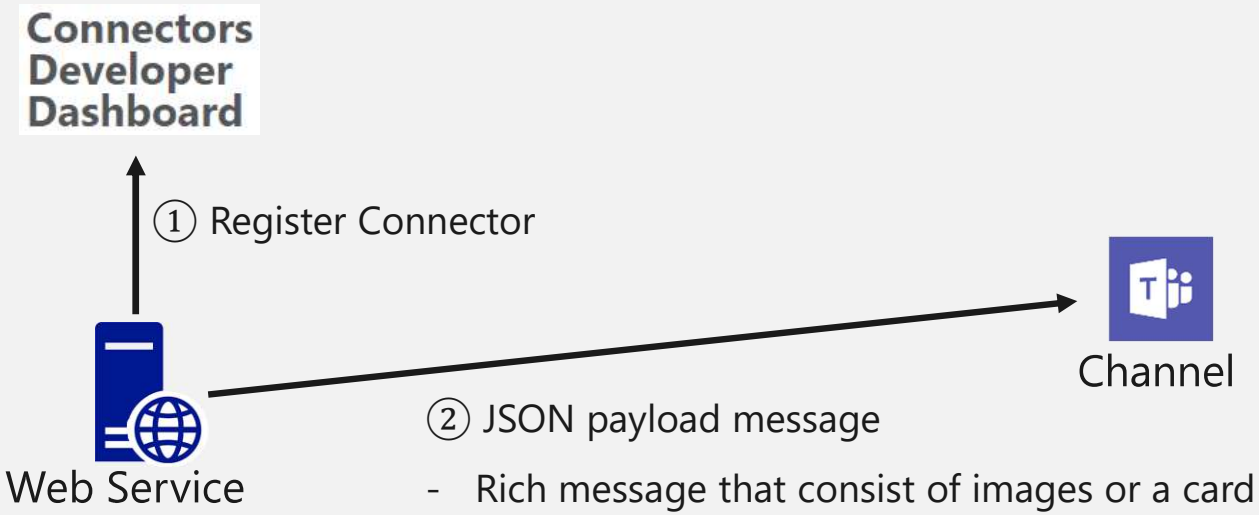


Demo : Add a RSS connector to a channel

7. Data provided by the connector is automatically posted to the channel



Demo : Develop custom connectors



The code does the following things:

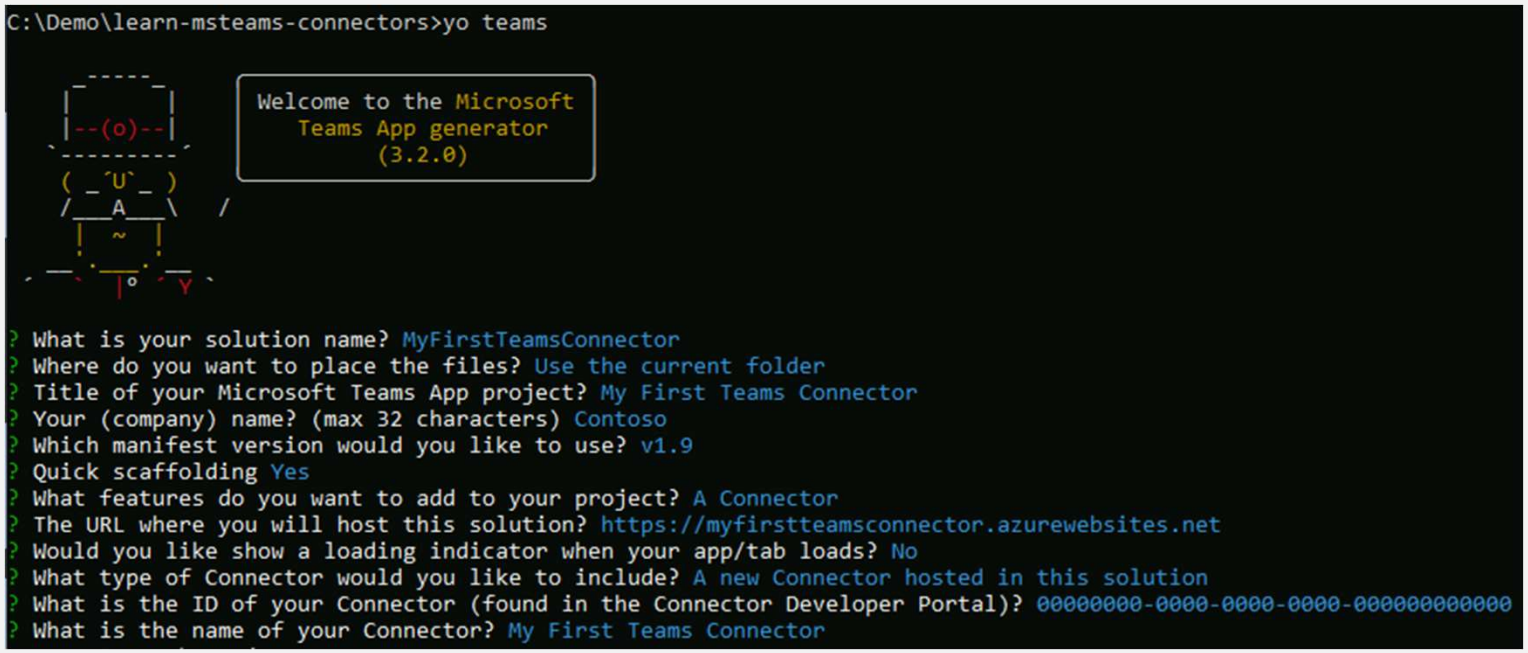
- Update the settings for the Connector in Microsoft Teams
- Submit an HTTP POST to the Connector's /api/connector/connect endpoint with a payload that contains necessary information the Connector web service needs to store

Demo : Develop custom connectors

- Create Teams app

  mkdir <learn-msteams-connectors>

  yo teams



Demo : Develop custom connectors

./src/manifest/manifest.json.

```
"connectors": [
  {
    "connectorId": "{{CONNECTOR_ID}}",
    "configurationUrl": "https://{{PUBLIC_HOSTNAME}}/myFirstTeamsConnector/config.html?name={loginHint}&tenant={tid}&group={groupId}&theme={theme}",
    "scopes": [
      "team"
    ]
  }
],
```

- This Connector ID can be found in the ./.env file that's used for development.
- Delete properties from manifest.json except "**connectors**"

  "configurableTabs": [], "staticTabs": [], "bots": [], "connectors": [], "composeExtensions": [],

Demo : Develop custom connectors

- Update setcolor() in useEffect() method

  ./src/client/myFirstTeamsConnector/MyFirstTeamsConnectorConfig.tsx.

```
microsoftTeams.settings.getSettings((settings: any) => {
    setColor(availableColors.find(c => c.code === settings.entityId));
});
```

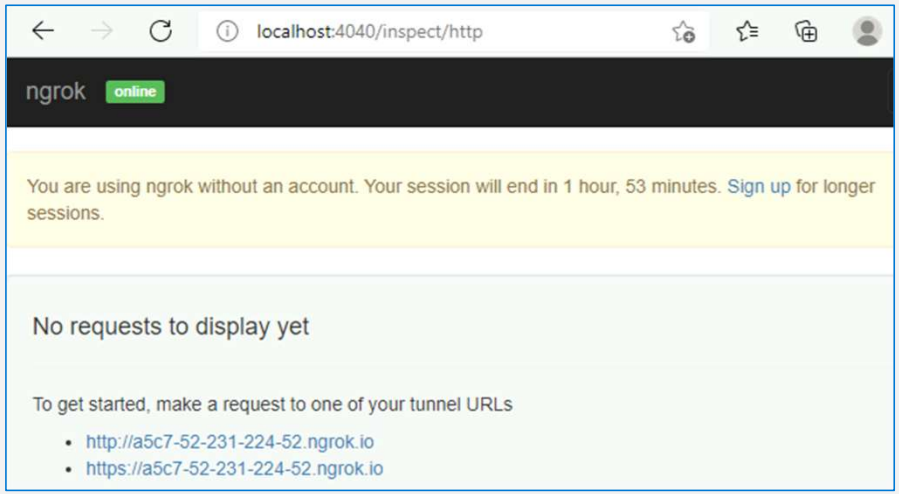
```
microsoftTeams.settings.getSettings((settings: any) => {
    setColor(availableColors.filter(c => c.code === context.entityId)[0]);
});
```

- gulp ngrok-serve

```
C:\Demo\learn-msteams-connectors>gulp ngrok-serve
[17:22:48] Found additional Yo Teams plugin: yoteams-deploy
[17:22:48] Using gulpfile C:\Demo\learn-msteams-connectors\gulpfile.js
[17:22:48] Starting 'ngrok-serve'...
[17:22:48] Starting 'start-ngrok'...
[17:22:48] [NGROK] starting ngrok...
[17:22:50] [NGROK] Url: https://a5c7-52-231-224-52.ngrok.io
[17:22:50] [NGROK] You have been assigned a random ngrok URL that will o
```

Browse http://localhost:4040,

Get dynamic URL by ngrok



Demo : Develop custom connectors

- Register an Office 365 Connector

  Open a browser and navigate to the Connectors Developer Dashboard:

  https://aka.ms/ConnectorsDashboard

Demo : Develop custom connectors

- Select New Connector.

  On the Register Connector page, complete the required fields

  Connector name:

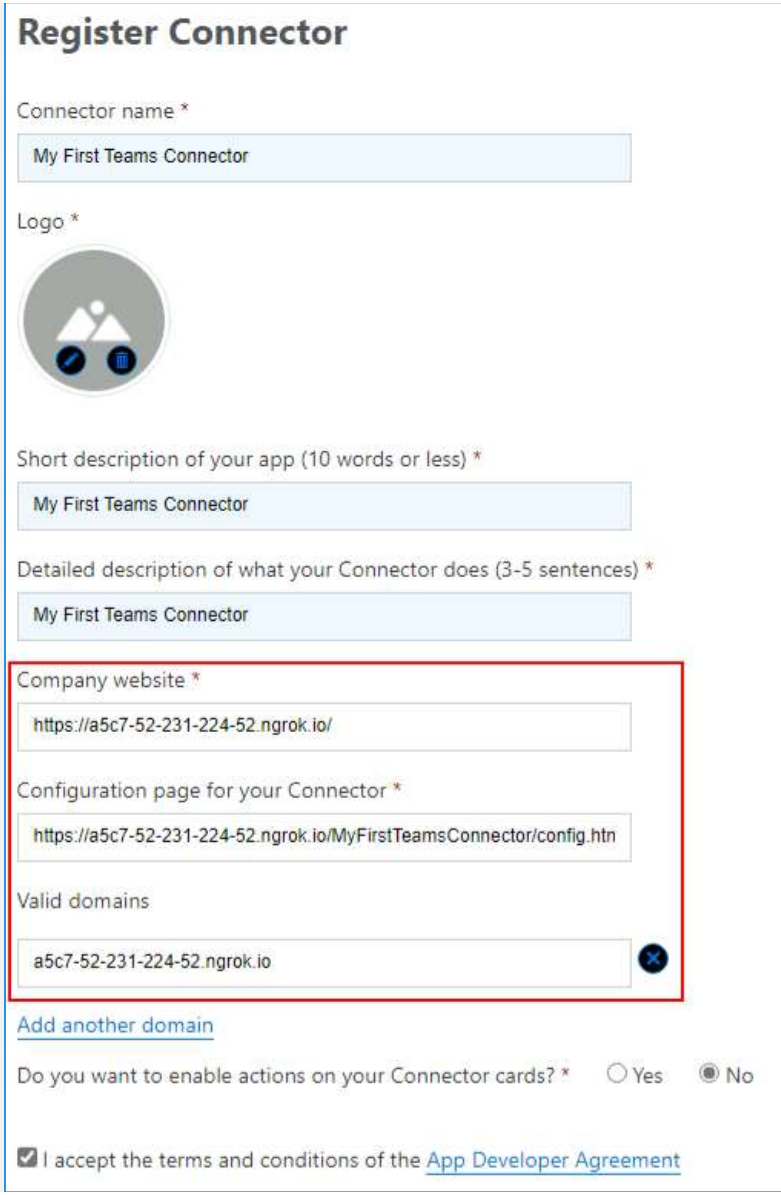
  My First Teams Connector

  Configuration page for your Connector:

  https://*REPLACE*.ngrok.io/MyFirstTeamsConnector/config.html

  Valid domains:

  REPLACE.ngrok.io

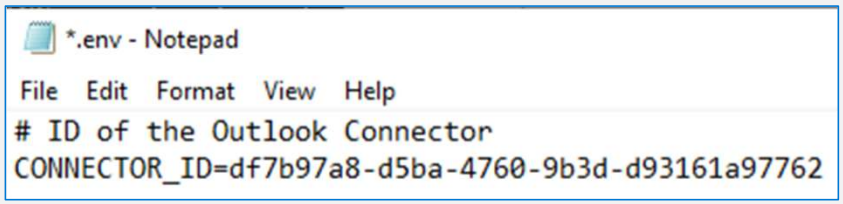


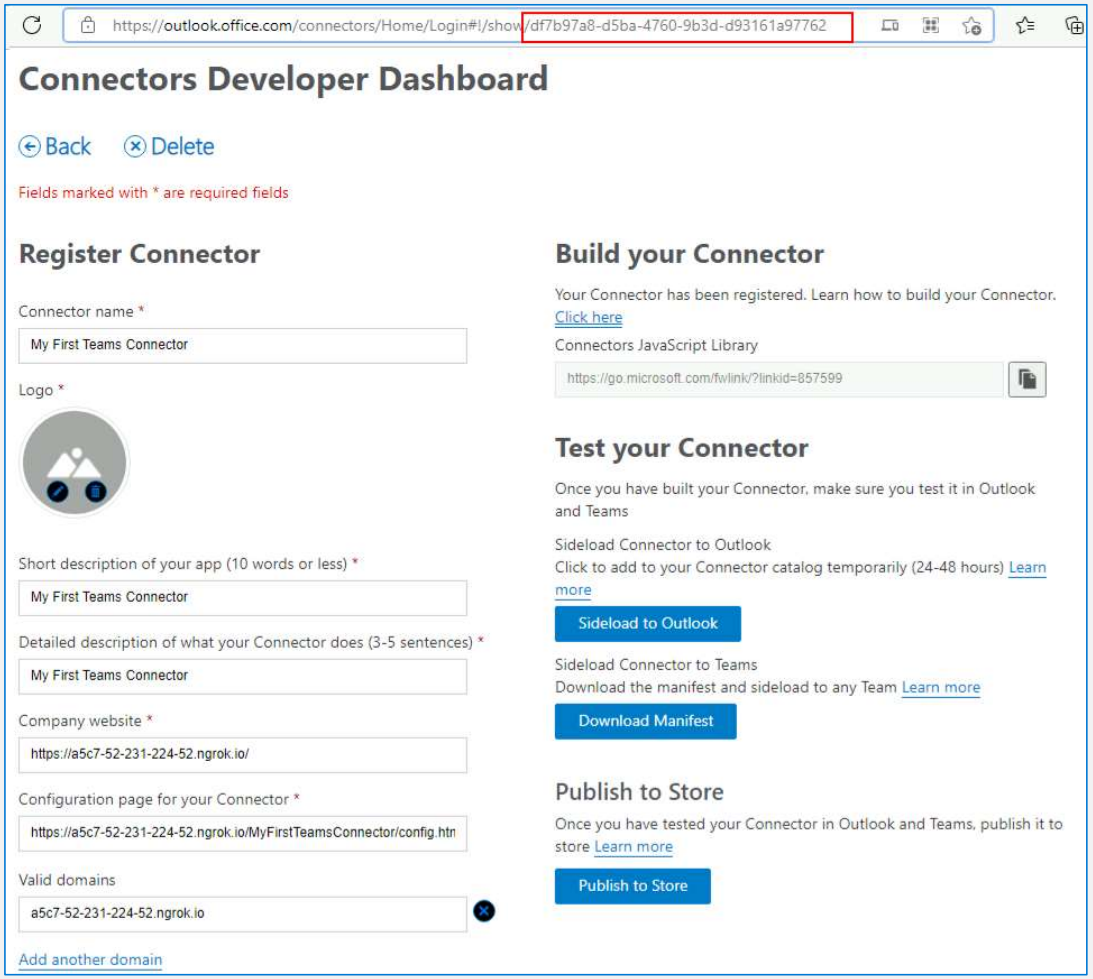
Demo : Develop custom connectors

After successfully registering your Connector, the Connectors Developer Dashboard page will display some additional sections.

While there's a button to Download Manifest for a custom Microsoft Teams app, we'll use the manifest created by the Yeoman Generator for Microsoft Teams.

This ID, a GUID, can be found in the URL of the updated page.





Demo : Develop custom connectors

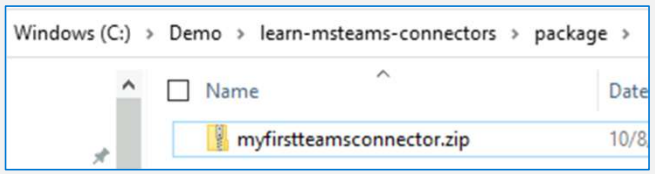
You need right Connector ID in **manifest.json** file. Two options :

- Update **.env** file, you'll re-start ***gulp ngrok-serve***.

  It'll generate new ngrok subdomain which would require update Connector's registration details.

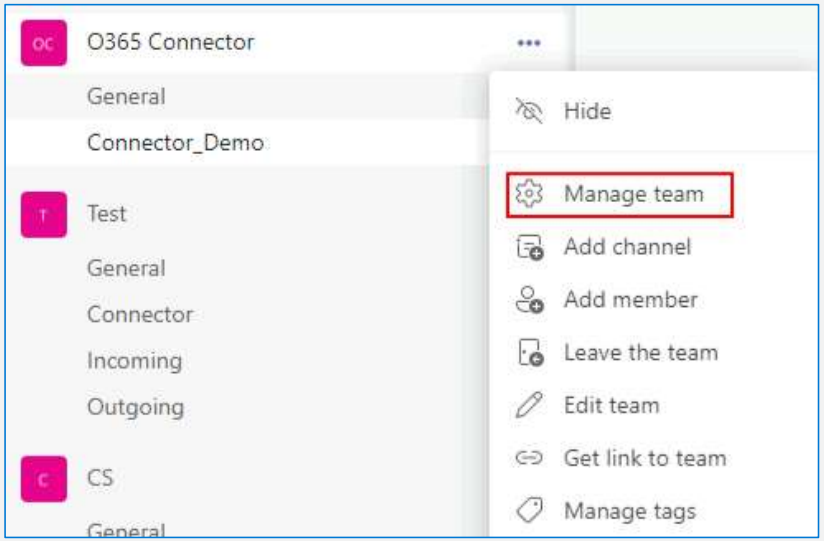
- Unzip package file in "**/package**" folder. And update connectors[0].connectorID property.

  Create a new app package ZIP with updated **manifest.json** file.

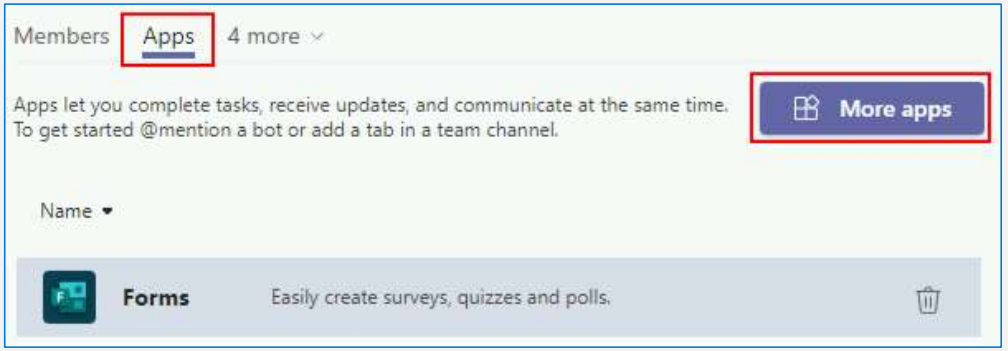


Demo : Develop custom connectors

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



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



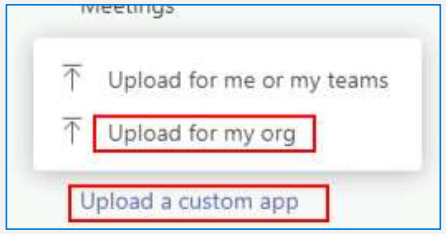
Demo : Develop custom connectors

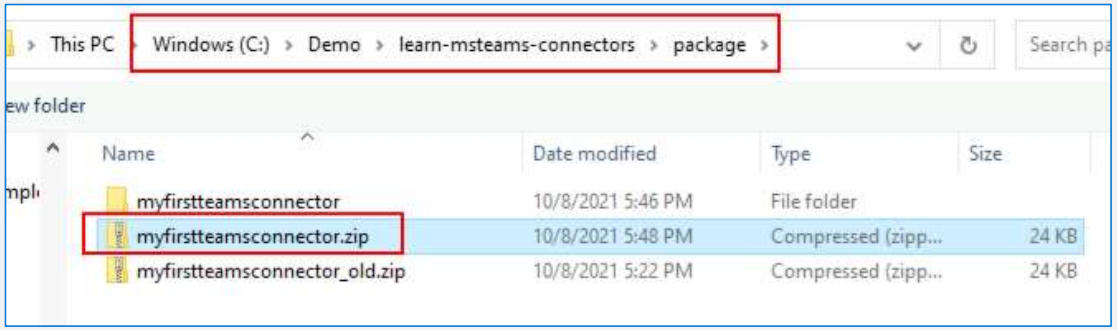
From the **Browse available apps and services**,

select the **Upload a custom app > Upload for my org** 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.

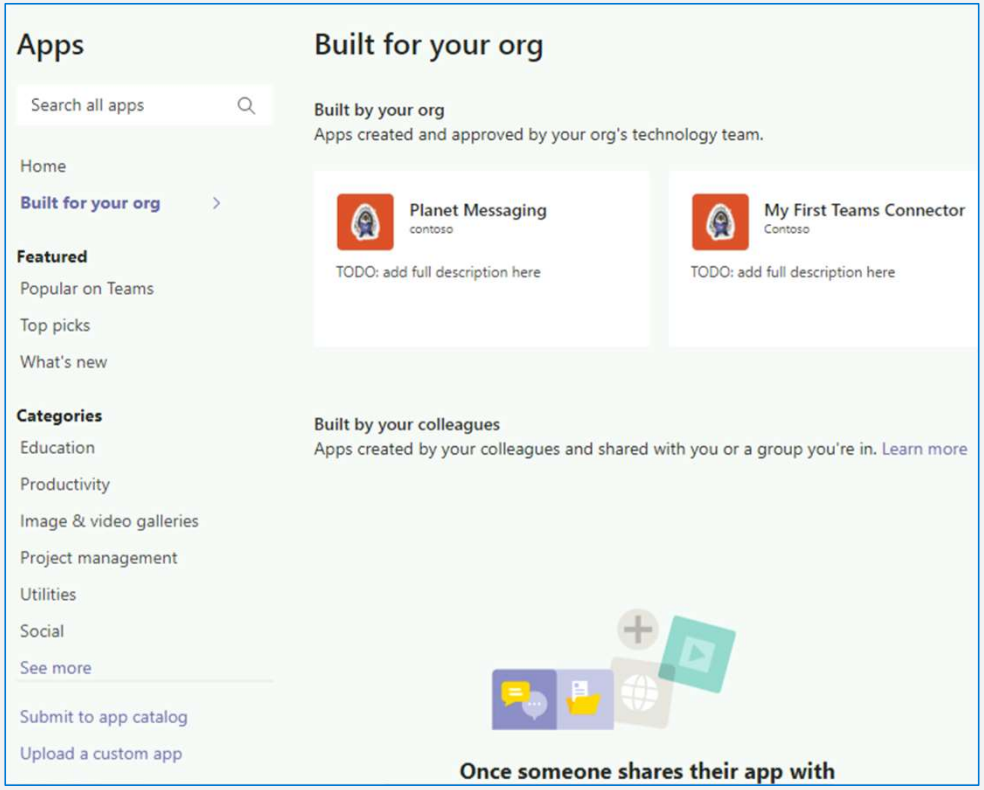




Demo : Develop custom connectors

After uploading the app,

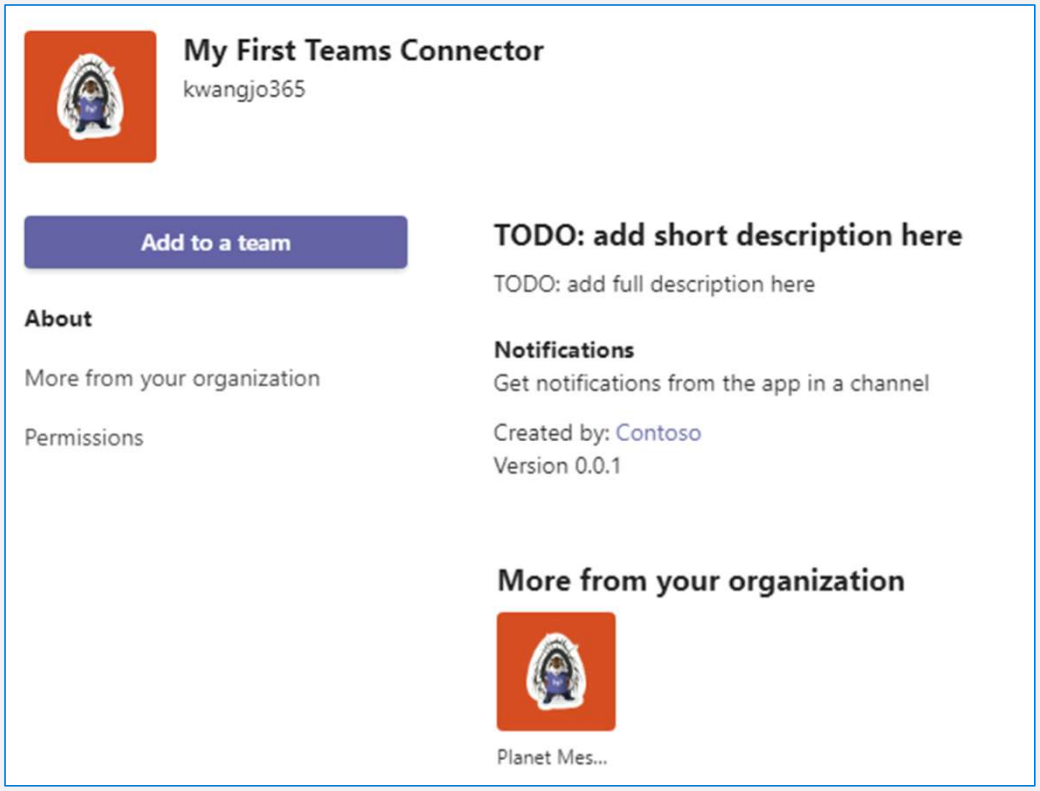
Microsoft Teams will display it on the list of apps installed under the **Build for [tenant]** category page:



Demo : Develop custom connectors

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.

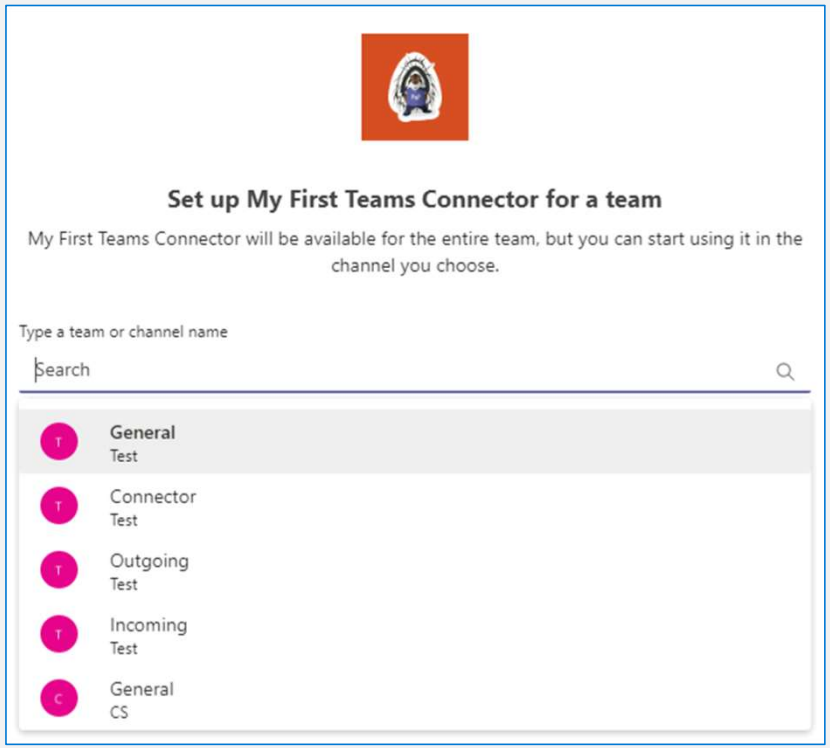


Demo : Develop custom connectors

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.

Then select **Set up a connector**.

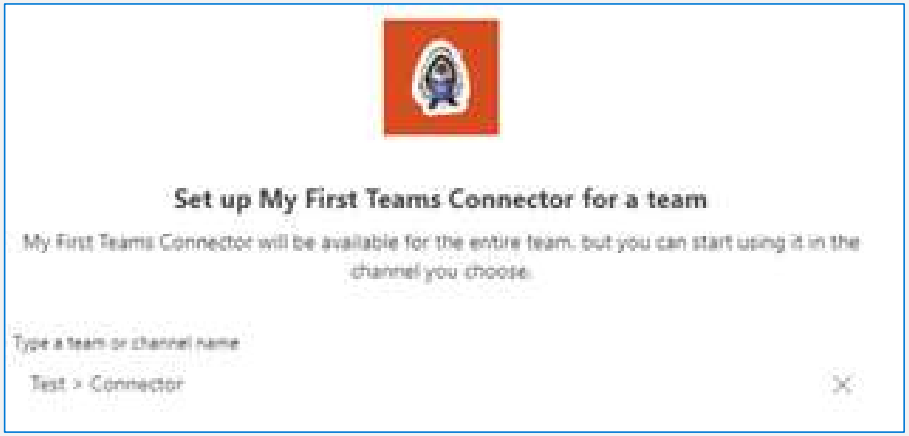


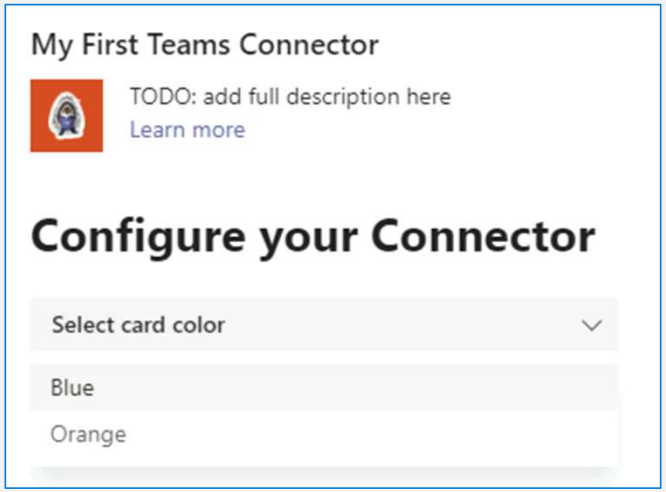
Demo : Develop custom connectors

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.





Demo : Develop custom connectors

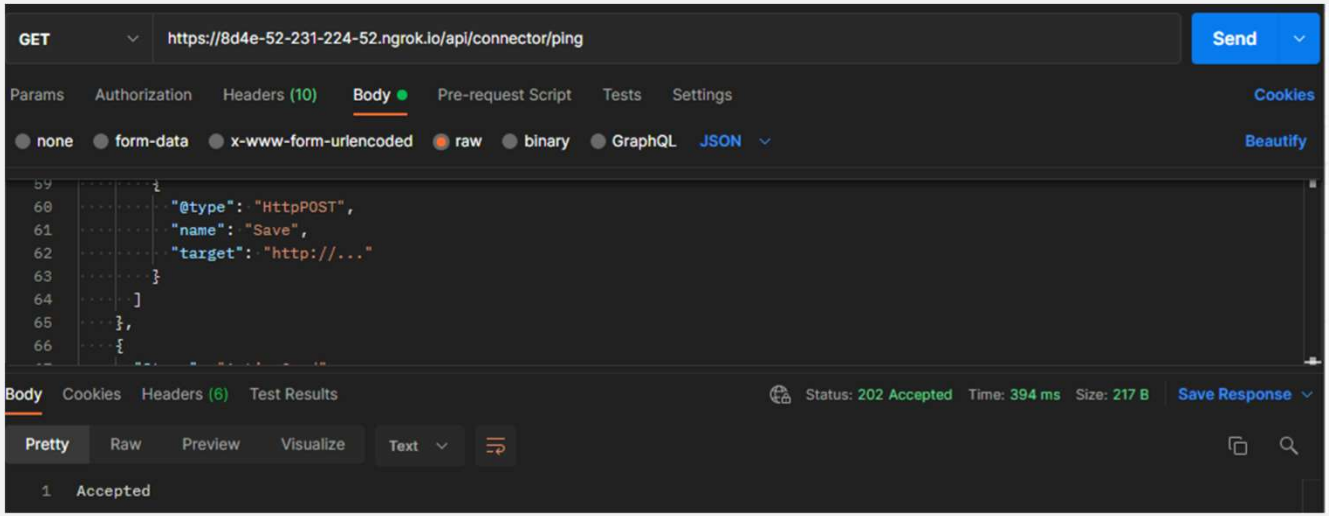
With the Connector saved, the next step is to see it post to a channel. Do this by submitting an HTTPS request to the web service's /ping endpoint.

Using the free tool Postman, 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

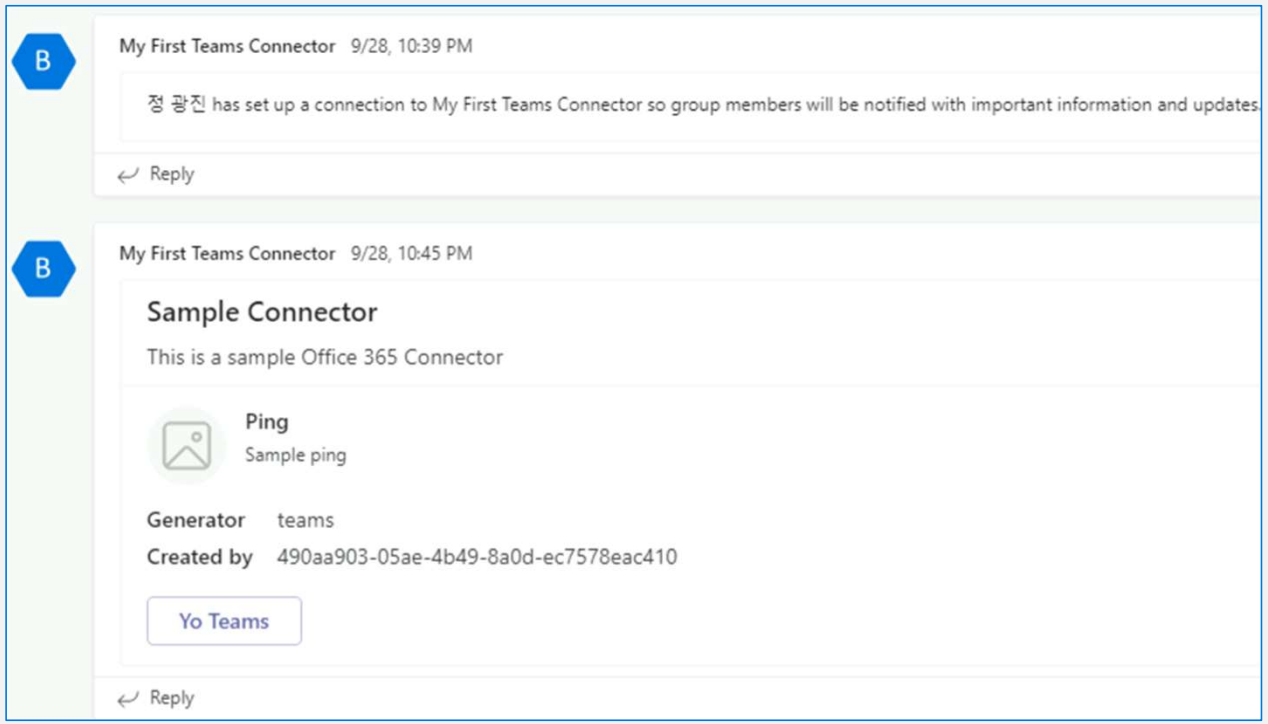
(make sure to update the URL of the image in the `sections[0].activityTile` property)



Demo : Develop custom connectors

Select the **Send** button in Postman.

When you go back to the channel, you'll see the card displayed as a message in the team:



05. Lab

Lab1. Create outgoing webhooks

https://docs.microsoft.com/en-us/learn/modules/msteams-webhooks-connectors/3-exercise-outgoing-webhooks

Lab2. Create incoming webhooks

https://docs.microsoft.com/en-us/learn/modules/msteams-webhooks-connectors/5-exercise-incoming-webhooks

Lab3. Create and add Office 365 Connectors to teams

https://docs.microsoft.com/en-us/learn/modules/msteams-webhooks-connectors/7-exercise-o365-connectors

Questions?

06. Appendix

Useful Link

- Microsoft Teams Dev Center | APIs and App Development
- Microsoft Teams Platform developer documentation - Teams | Microsoft Docs
- Microsoft Teams 개발 앱 - 연결 - Learn | Microsoft Docs
- Microsoft Teams의 앱, 봇 및 커넥터 - Microsoft Teams | Microsoft Docs
- TrainingContent/Teams at master · OfficeDev/TrainingContent · GitHub
- GitHub - OfficeDev/Microsoft-Teams-Samples: Welcome to the Microsoft Teams samples repository. Here you will find task-focused samples in C#, JavaScript and TypeScript to help you get started with the Microsoft Teams App!
- Adaptive Cards

Connector Cards – Design Guidelines

Sections

Use to logically group data together

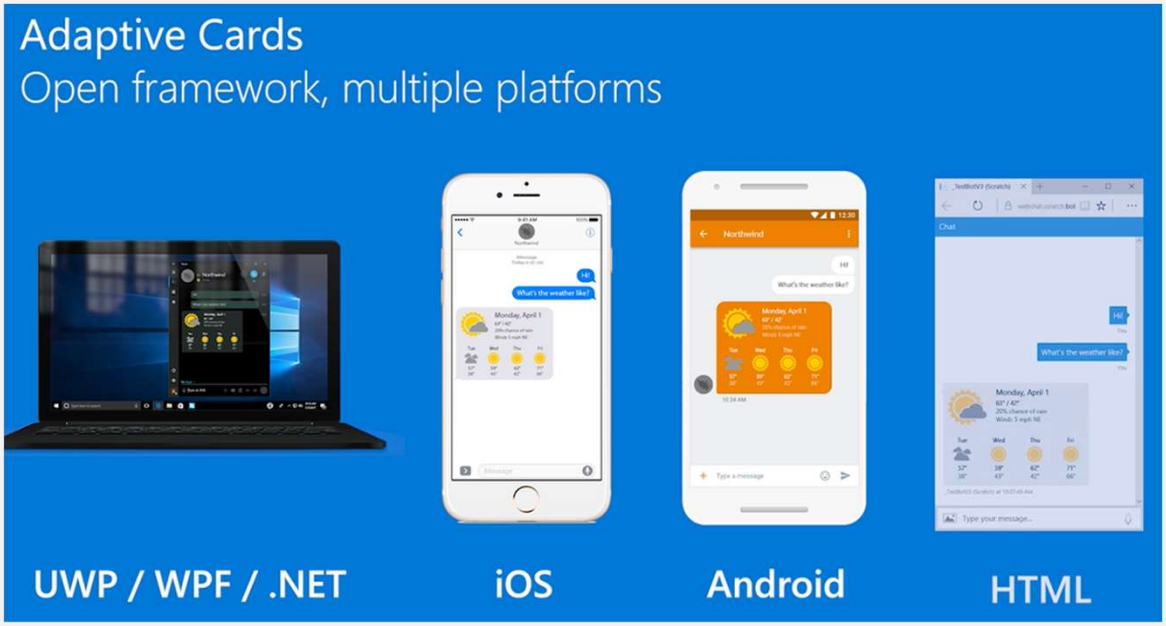
Don’t include more than 10

Adaptive Cards

Adaptive Cards are platform-agnostic snippets of UI, authored in JSON, that apps and services can openly exchange. When delivered to a specific app, the JSON is transformed into native UI that automatically adapts to its surroundings. It helps design and integrate light-weight UI for all major platforms and frameworks.

https://adaptivecards.io/

https://github.com/microsoft/AdaptiveCards/



Development Applications

Jira Software	Incoming Webhook	Tabs	Bots	Messaging	Personal App
Jira Core	Incoming Webhook	Tabs	Bots	Messaging	Personal App
Confluence		Tabs		Messaging	
Bitbucket	Incoming Webhook	Tabs	Bots	Messaging	Personal App
Jenkins	Incoming Webhook				
New Relic	Incoming Webhook				
LaunchDarkly	Incoming Webhook