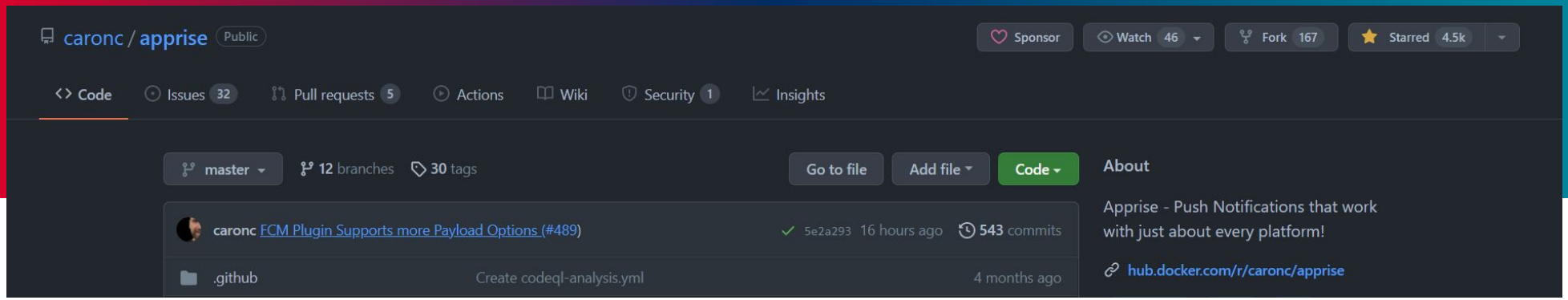
Apprise - your push messaging musketeer: one for all (messenger services)

JS Jörg Schultze-Lutter | Last modified: 13.04.2022 | 4 minutes read



Apprise allows you to send a notification to almost all of the most popular notification services available to us today

What is Apprise?

Apprise is an open source application that allows you to **send push messages to more than 70 services** (Messenger, SMS, Email, Home Assistant, ...) **without having to integrate the individual native APIs of these services on your end**. Messages can be sent to **multiple messaging services at the same time**, so if a single messaging service fails, the message is still delivered via backup services.

Apprise was written in Python and **can either be used as a standalone program** (command line) **or** integrated into existing code - there is also a Docker image available. All service-specific credentials are either stored in Apprise-specific configuration files or can be passed to Apprise as parameters. All **credentials are protected** and do not appear in log files.

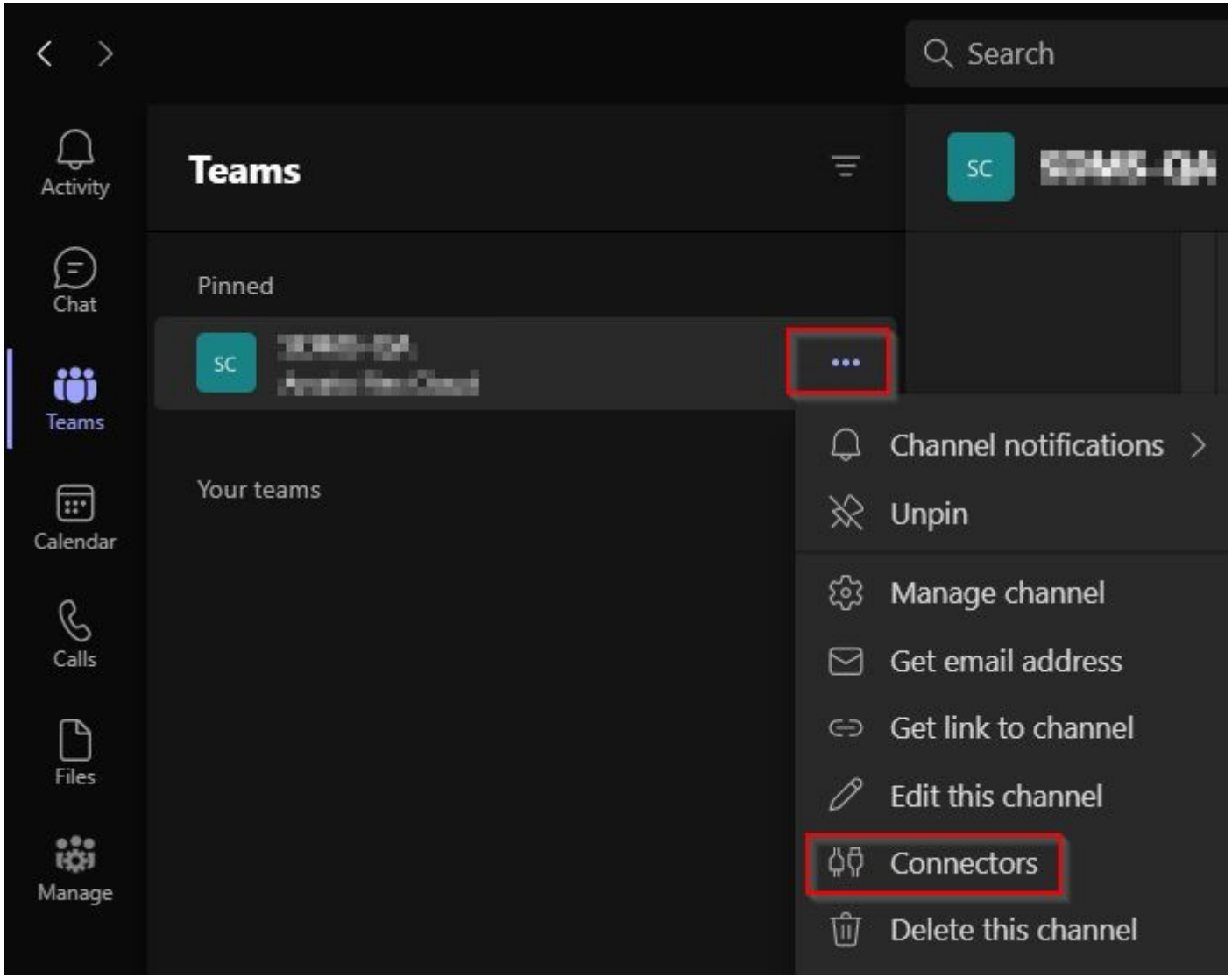
Depending on the capabilities of the messaging service, Apprise **supports HTML and/or text messages**. In addition, **users can send file attachments, such as log files or images**, if supported by the messaging service. To prevent the messaging process from accidentally blocking your code, Apprise **uses asynchronous processing**.

Example configuration of Apprise and a Microsoft Teams channel

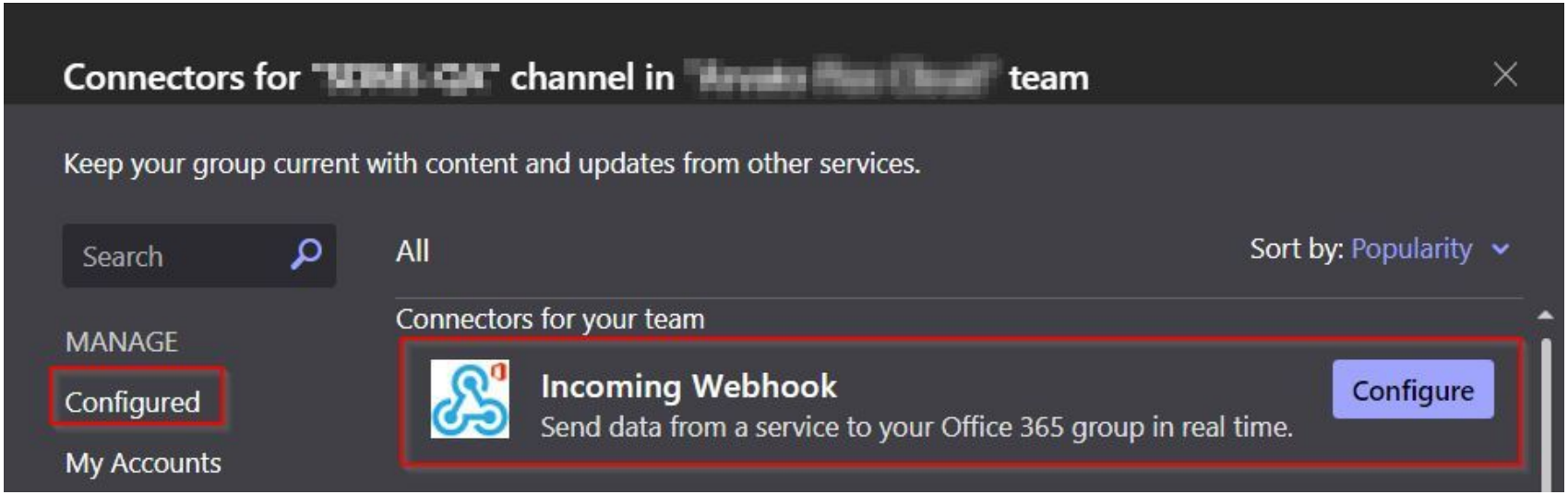
In the following example I will create a sample configuration of Apprise for sending a message to a Microsoft Teams channel. Keep in mind that each Apprise-supported messenger service has its own individual configuration documentation, which you can access either via the main page of the Apprise project or via its wiki. Thus, we first select the Microsoft Teams configuration documentation for Apprise in the wiki, where we can get generic information about the plugin's configuration, messaging capabilities, etc.

Creating the Webhook in Microsoft Teams

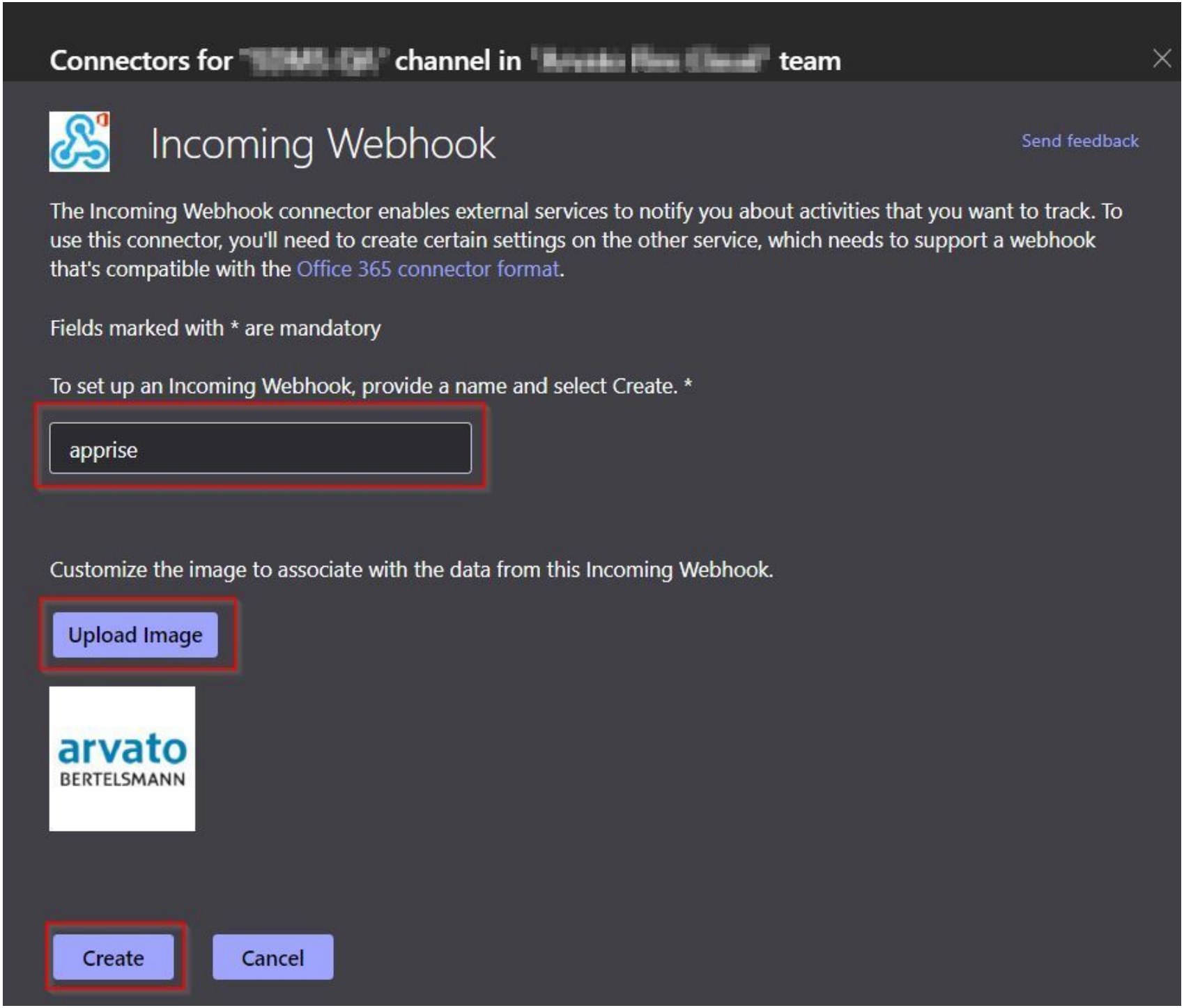
In order to be able to configure Apprise for Microsoft Teams, we first need to set up a so-called Incoming Webhook. This Webhook can be configured in Microsoft Teams, asssuming that you are an administrator of the channel in question. To do this, first go to your channel in Microsoft Teams and then click on the three dots behind the name of your channel, and select the menu item ***Connectors***.



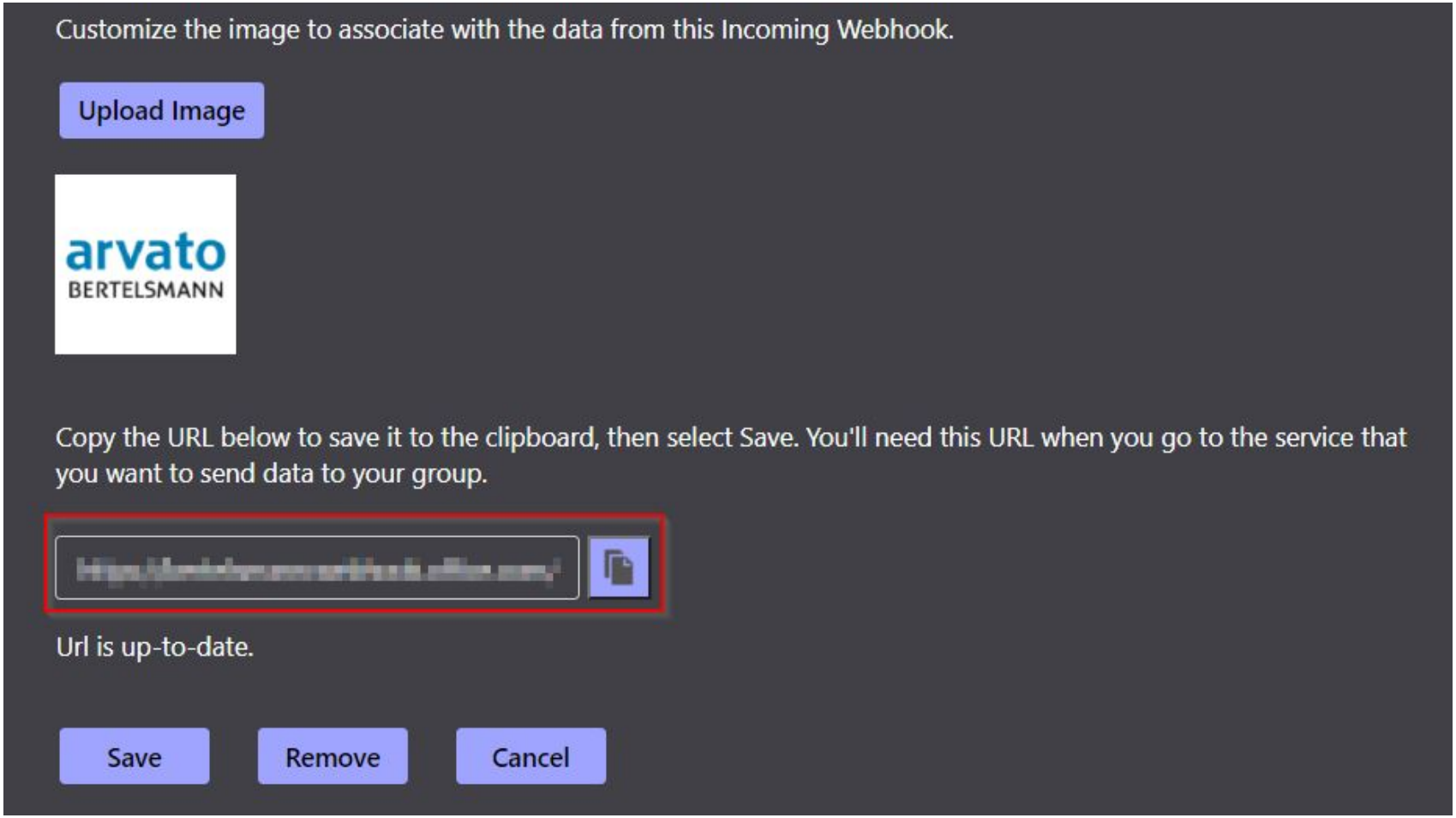
To create a new Webhook, click the ***Incoming Webhook*** item's ***Configure***button. If you need to edit this webhook later or look up the Webhook's URL again, select ***Manage /Configured*** instead.



The data required for creating a webhook is rather limited. You just have to specify a name and an optional icon for your webhook.


Connectors for [redacted] channel in [redacted] team
Incoming Webhook
Send feedback
The Incoming Webhook connector enables external services to notify you about activities that you want to track. To use this connector, you'll need to create certain settings on the other service, which needs to support a webhook that's compatible with the Office 365 connector format.
Fields marked with * are mandatory
To set up an Incoming Webhook, provide a name and select Create. *
apprise
Customize the image to associate with the data from this Incoming Webhook.
Upload Image
arvato
BERTELSMANN
Create
Cancel

Once the webhook is created, create a copy of the link and close this dialog.



Your webhook will look something like this:

> https://bcp.webhook.office.com/webhookb2/f13637ba-adcb-4312-8966-a1b2c3ddefd8@d8aa127b-66d4-4ff2-1234-abbf1234daab/IncomingWebhook/09abdfba21b046d7a172d5133199293f/9b22a5e7-7dda-4bb3-1317-82b13e9ff09

We now need to convert this webhook into the Apprise-specific format so that it can be used by Apprise's Microsoft Teams plugin. To do this, we follow the plugin's wiki documentation.

The resulting link looks like this:

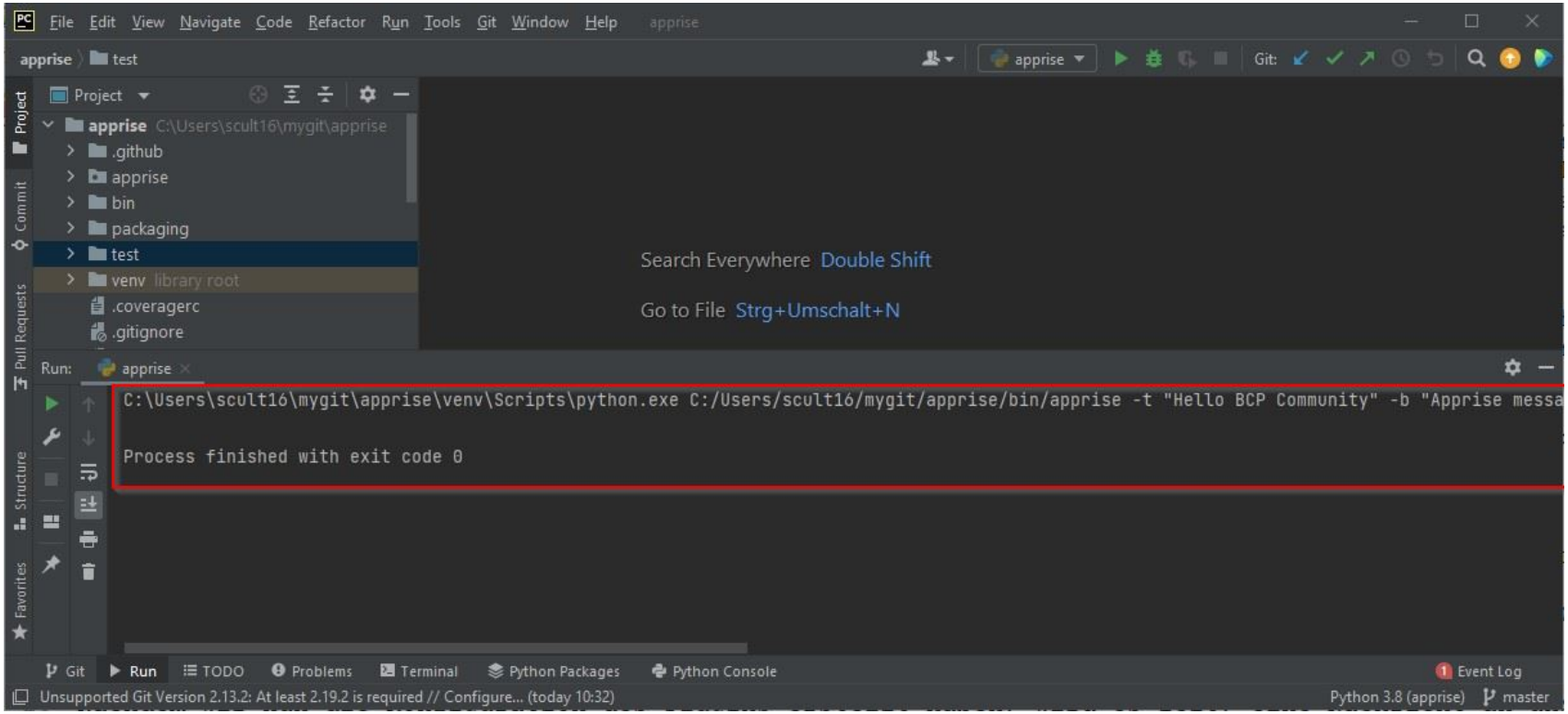
```
msteams://bcp/f13637ba-adcb-4312-8966-a1b2c3ddefd8@d8aa127b-66d4-4ff2-1234-
abbf1234daab/09abdfba21b046d7a172d5133199293f/9b22a5e7-7dda-4bb3-1317-82b13e9ff09
```

Time for a test run!

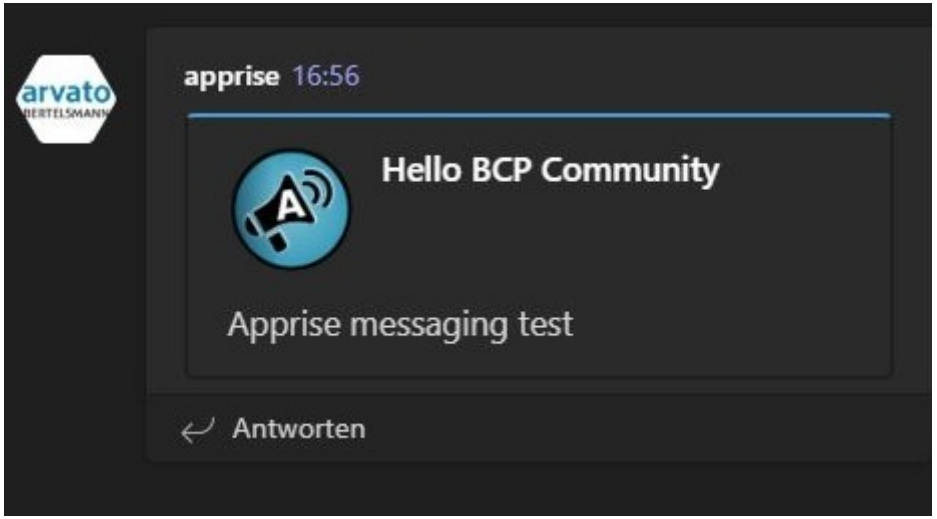
Now that we've finished configuring the Apprise plugin, we can finally send a message to our Microsoft Teams channel. To do this, we pass a *message body* as well as an optional *message title* to Apprise in addition to the configuration we just created:

```
apprise -t &amp;quot;Hello BCP Community&amp;quot; -b &amp;quot;Apprise messaging test&amp;quot;
msteams://bcp/f13637ba-adcb-4312-8966-a1b2c3ddefd8@d8aa127b-66d4-4ff2-1234-
abbf1234daab/09abdfba21b046d7a172d5133199293f/9b22a5e7-7dda-4bb3-1317-82b13e9ff09
```

In the following example, I use PyCharm to call the Apprise application/API, but this call could just as easily have been made via command line or else by calling Apprise as a library from your code.



In case of success, Apprise should then return a return code of 0 and the message should be visible in your Microsoft Teams channel.



Create your own Messenger plugin

If your favorite messenger service is not available in Apprise, it is easy to implement new plugins, provided of course that your messenger service offers an appropriate API.

Preferably, check if an existing Apprise plugin service matches your messenger's feature set in message format and parameters and then use this plugin as a template for your new messenger service. Alternatively, you can take a look at the developer wiki, where the basics of creating your own plugin are discussed.

For the creation of the actual plugin the following files / directories might be relevant:

- The `apprise/apprise/plugins` directory. This is where you keep your messenger plugin. You will also find all existing plugins in this directory.
- The file `apprise/apprise/utils.py` defines regular expressions that are used to validate the input parameters of your later plugin. Normally you will not have to make any changes here.
- Additionally, you should provide a test of your plugin under `apprise/tests`.

Please note that your plugin code must provide Python2 backwards compatibility; this requirement will be enforced as part of the pull request's validation process.

Finally, when creating new plugins, I always recommend to provide a corresponding wiki page, which shows how the plugin can be used and which restrictions like text format and length it has.

Questions / Comments

If you are interested in creating your own plugin and contributing to Apprise's code base, feel free to drop me a line, check out **my contribution to Apprise's codebase**, or get in touch with Apprise's core developer **Chris**.

The opinions and information stated in this article are personal to the individual author and do not necessarily represent Bertelsmann.

About the Author

Jörg Schultze-Lutter

Arvato

Tags