Software Evolution - Practical Session: Building a GitHub Bot

Guest Lecturer: Mairieli Wessel, Radboud University, The Netherlands

This practical session aims to familiarize yourself with the development of bots for managing GitHub repositories and explore the benefits and challenges of developing and using GitHub bots.

GitHub bots are applications that automate workflow activities on GitHub using webhooks and the GitHub API. Bots can do many different things, such as automatically responding to users, applying labels, closing issues, creating new issues, and merging pull requests.

In this document, you will find useful resources and detailed instructions on how to prepare the environment (in **Step 0**), develop the bot (**Step 1**), and do the assignment exercises (**Step 2**).

Resources

You can find relevant information in the GitHub API v3 documentation.

→ Events and payloads: https://docs.github.com/en/developers/webhooks-and-events/webhooks/webhook-events-and-payloads

Step 0: Preparations

Before coming to the practical session, please do the following:

- Make sure you have **Python 3.9.X** installed on your machine.
- Make sure you also have **npm** installed.
- Create a **GitHub account** if you do not already have one yet.

There are other **important resources** that we are going to use throughout the session:

- **Flask** as our Python webserver
  - Installation: `pip install -U Flask`
  - Flask documentation: https://flask.palletsprojects.com
- **PyGitHub** to communicate with GitHub
  - Installation: `pip install PyGitHub`
  - PyGitHub documentation: https://pygithub.readthedocs.io
- **cryptography** for decrypting our GitHub App certificates
  - Installation: `pip install cryptography`
  - cryptography documentation: https://cryptography.io
- **smee.io** to receive payloads from GitHub
  - Installation: `npm install --global smee-client`

- After the installation, validate it by running: `smee --version` . If the *smee* version appears, you are good to go!

Step 1: Bot development — Greeting developers

1.1 Create two new repositories on GitHub

You need to create **two repositories on GitHub** to complete the practical session.

→ The **first repository** is needed to hold the codebase for the bot. This is where you will push the source code. Fork the following repository, which contains the source code of the bot presented in the practical session:

https://github.com/mairieli/webservice-bot-umons

→ The **second repository** will be where the bot will be installed and tested. Your bot will interact with this repository. In the real world, this will be the project that you're managing. I will be using my personal repo, https://github.com/mairieli/test-bot-umons.

1.2 Create a Webhook Payload Delivery

→ Go to smee.io and click on "**Start new channel**"

→ Copy the **WebHook Proxy URL** and save it for later use

1.3 Create a GitHub App

→ Log in to your GitHub account, click on **your account icon** → **Settings** → **Developer settings**

→ Go to **GitHub Apps**, and click on the **New GitHub App** button at the top right

→ Give your app a **name** (**GitHub App name**) and a **homepage** (**Homepage URL**)

→ Go to the **Webhook section:** in the **Webhook URL**, copy and paste the URL that you got from smee.io

→ Go to the ***Repository permissions*** section: here, we will define what permission our bot will need when someone installs it on its repository. Our bot needs (so far!) permission to read & write a comment on a pull request, so you need to add ***Access: Read & Write*** under the ***Pull requests*** section and the ***Issues*** section

→ Go to ***Subscribe to events*** and check the ***Pull request*** and ***Issues*** checkbox. It will tell the bot to send us an event payload when a pull request and issue events are created

→ Click on the ***Create GitHub App***

→ After you create the ***GitHub App,*** please save the ***App ID*** for later use

1.4 Install the bot

→ Go to **your account icon** → **Settings** → **Developer settings** → **GitHub Apps** and then select your bot app.

→ Click on **Install App** and install it on your second repository

1.5 Clone the bot project

→ Clone to your machine the repository you have forked (**Step 1.1**)

→ Run `pip install -r requirements.txt`

→ Change line 7 with your App ID (as Integer)

→ In line 10, we read the certificate for the bot. We need to create a private key to authenticate with GitHub. Go back to the bottom of the bot app page; you will find the ***Private keys*** section, click on ***Generate a private key***, and it will create and download a ***.pem file*** with the private key inside

1.6 Redirect the payload from smee.io to your bot

→ Run the following command:

```
smee -u https://smee.io/<YOUR CHANNEL ID> --port 5000
```

1.7 Run the webservice

→ Run `python3 app.py`

Step 2: Assignment

2.1 Exercises

Using the source code presented in the practical session as a baseline, implement the following bot features:

Exercise 1. Apply labels to issues.

Each time someone opens an issue, the bot automatically applies a label. This can be a “pending” or “needs triage” label.

As for the example presented in the lecture, you will want to subscribe to the **issue** event, specifically when the action to the event is **opened**.

For reference, the relevant GitHub documentation for references is here: https://developer.github.com/v3/issues/#edit-an-issue

Relevant PyGitHub documentation: https://pygithub.readthedocs.io/en/latest/github_objects/Issue.html#github.Issue.Issue.add_to_labels

Exercise 2. Say thanks when a pull request has been merged.

Make the bot post a comment to say thanks whenever a pull request has been merged.

For this case, you will want to subscribe to the **pull_request** event, specifically when the action to the event is **closed**.

For reference, the relevant GitHub API documentation for the pull request event is here: https://docs.github.com/en/webhooks-and-events/webhooks/webhook-events-and-payloads?actionType=closed#pull_request

Note: A pull request can be closed without it getting merged. You will need to find out how to determine whether the pull request was merged or closed.

Exercise 3. Automatically delete a merged branch.

Automate even more tedious tasks whenever a pull request has been merged. Make the bot automatically delete a merged branch. The branch name can be found in the pull request webhook event.

As for the previous exercise, you will want to subscribe to the **pull_request** event, specifically when the action to the event is **closed**.

For reference, the relevant GitHub documentation for references is here: https://docs.github.com/en/rest/git/refs

Relevant PyGitHub documentation: https://pygithub.readthedocs.io/en/latest/github_objects/GitRef.html#github.GitRef.GitRef

Exercise 4. Prevent merging of pull requests with "WIP" in the title.

Make the bot set a pull request status to **pending** if it finds one of the following terms in the pull request titles: “wip,” “work in progress,” or “do not merge.” Developers might update the pull request title at any time, so consider it and update the pull request status accordingly. For example, once the pull request is updated and the term removed, set back the pull request status to **success**.

For this case, you will want to subscribe to the **pull_request** event, specifically when the action to the event is **edited**.

For reference, the relevant GitHub documentation for the commit statuses is here: https://docs.github.com/en/rest/commits/statuses

Examples using PyGitHub: https://pygithub.readthedocs.io/en/latest/examples/Commit.html#create-commit-status-check

Note: Commit statuses can determine the overall status of a pull request. You must find out how to change the pull request status based on a commit status. Read the documentation for reference.

2.2 Assignment report

In a short report document, provide the following:

1. The URL of the public GitHub repository that contains the source code of all implemented features of the bot (outcomes of exercises 1, 2, 3, and 4).
2. For each exercise, include screenshots showing the successful bot interaction (i.e., bot outcome) on your testing repository.

2.3 Bonus exercise

Automatically overwrites the pull request status.

Implement a new feature for the bot to interact with a developer. Have the bot react to a developer commenting “@mons-bot ready for review” in a pending pull request. The bot overwrites the pending status regardless of the pull request title content.

Note: Replace “@mons-bot” in the command that calls the bot with the name of the bot you registered at GitHub.com.