Eskimon



Zeste de savoir

# Faire un bot Discord simple avec les webhooks

15 juin 2020

# Table des matières



Si vous vous promenez sur les réseaux de discussions de Discord, vous avez sûrement remarqué que ces derniers sont peuplés de bots qui postent des informations diverses et variées.

Récemment, Zeste de Savoir a aussi eu le droit à son propre serveur Discord non-officiel . C'était l'occasion rêvée de voir naître un tutoriel qui parle des bots.

Ce petit tutoriel va donc vous permettre de créer un petit bot simple qui se contentera de poster un message dont le contenu sera personnalisable à loisir. C'est tout. Pour le réaliser, tout sera écrit en python 3 *vanilla*, donc sans aucune bibliothèque tierce. Nous allons ici voir l'utilisation des *webhooks* dans le cadre de Discord, mais cette technique marche évidemment avec plein d'autres services qui le permettent (comme IFTTT par exemple).

> *i* Le bot que nous allons créer ne sera pas très évolué. Il sera uniquement en mesure de poster des messages mais ne pourra pas lire le contenu des canaux de communication. En effet tout cela est limité par la technique utilisée ici, les ***webhooks*** Discord.

## 1. La technique des webhooks

### 1.1. Présentation

Comme dit dans l'introduction, ce tutoriel repose sur l'utilisation des *webhooks* pour assurer la communication entre notre bot et Discord.

Mais un *webhook* qu'est-ce que c'est ?

D'après wikipédia (en anglais), un *webhook* est une méthode pour manipuler le comportement d'une page via l'utilisation d'une « fonction de rappel » (*callback*). Plus concrètement ici, un *webhook* nous permet de poster un contenu (« manipuler une page ») via l'envoi d'un message (notre fonction de rappel).

De manière très simple, la situation se présente ainsi :

— Discord nous fournit une URL reliée à un salon ;

— Notre bot fera une requête `POST` sur cette URL avec le message à poster ;
— Discord affichera alors le message.

Très concrètement, la seule chose à faire sera donc de faire une requête `POST` sur une URL précise. *Easy* ! 🍊

## 1.2. Obtenir l'URL du webhook

Pour commencer, il va tout d'abord falloir obtenir notre URL de communication ! Pour cela, cliquez sur le nom de votre serveur puis allez sur l'option « Paramètres du serveur ». Sur la capture ci-dessous le nom du serveur est entouré.

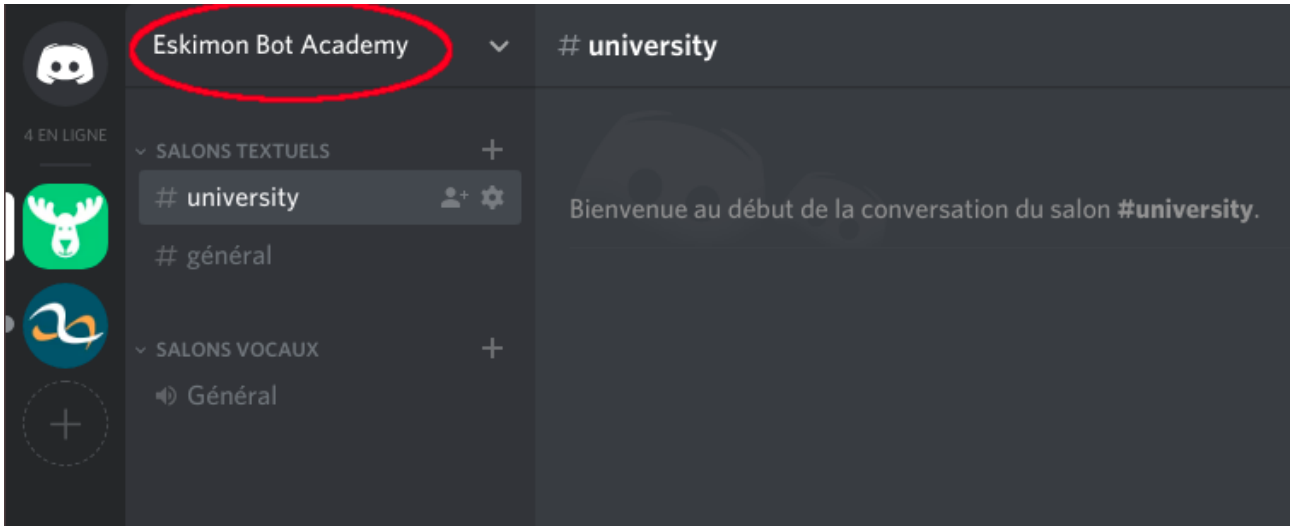


FIGURE 1.1. – Emplacement des paramètres du serveur

Une fois cela fait, sélectionnez l'option « *Webhook* » à gauche puis cliquez sur « Créer un *webhook* ».

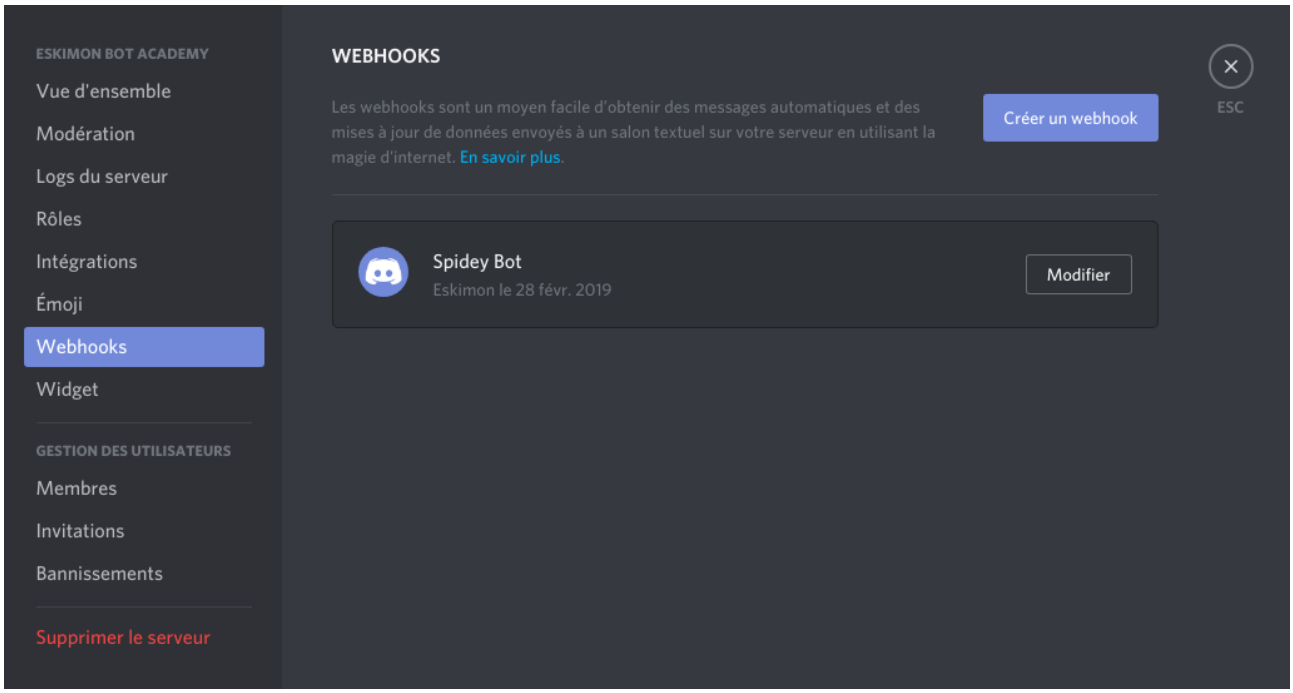


FIGURE 1.2. – L'option Webhook

Enfin, donnez un petit nom et une bouille sympathique à votre bot puis assignez-le au canal de discussion qui vous convient :

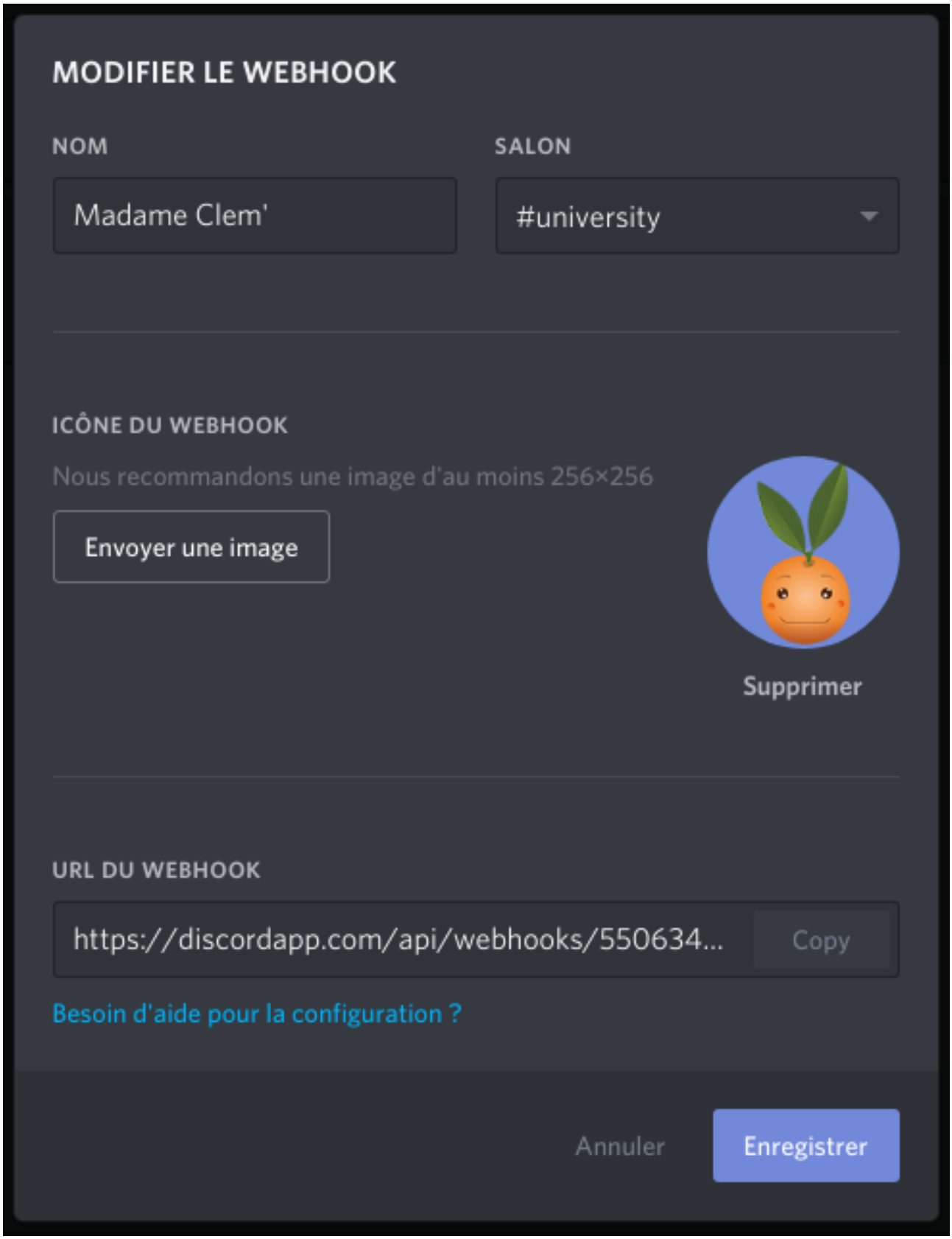


FIGURE 1.3. – Création d'un bot

Dernière chose, copiez l'adresse fournie en bas quelque part, ce sera celle qui nous servira justement à communiquer !

Et voilà, tout est prêt du côté de Discord, place au code maintenant !

## 2. Émettons des messages

Passons maintenant à la partie amusante, l'émission de messages depuis notre bot vers le canal Discord.

Comme dit plus tôt, l'objectif sera de simplement faire une requête `POST` vers l'URL que nous venons d'obtenir. Pour générer cette dernière, nous allons donc utiliser la bibliothèque `request` incluse nativement dans python :

```
1 from urllib import request
2 from urllib.error import HTTPError
```

(J'en profite aussi pour importer les exceptions que cette dernière peut générer)

Maintenant, il va falloir faire un message. Mais pas n'importe comment ! En effet, il faut prendre en compte ce que nous raconte la documentation de Discord au sujet des *webhooks* . Ainsi, on apprend que le message doit contenir un *payload* **json** avec au choix :

- Un champ `content` qui représente un simple message pouvant faire jusqu'à 2000 caractères ;
- **OU** un champ `embeds` contenant une liste d'objets `embed` ;
- **OU** un champ `file` qui contiendra un fichier à envoyer (la *payload* devra alors être au format `multipart/form-data` avec un champ `payload_json`).

Je vais uniquement parler des 2 premiers cas, le troisième étant plus particulier et tout de même simple une fois le principe compris.

### 2.1. Envoyer un message simple

Nous allons commencer par envoyer un message tout simple qui ressemblera au suivant :


FIGURE 2.4. – Un message tout simple

Pour cela, on commence par créer notre dictionnaire qui sera notre futur objet json. On fait donc un simple dictionnaire python avec une seule clé : `content`.

```
payload = {
    'content': "Salut les z'agrumes ! Ça zeste ?"
}
```

Maintenant, il s'agit de fournir cela proprement à `request` pour pouvoir l'émettre.

```
# À remplacer par celle que vous avez obtenu précédemment
WEBHOOK_URL = 'https://discordapp.com/api/webhooks/xxx/yyy'

req = request.Request(url=WEBHOOK_URL,
                      data=json.dumps(payload).encode('utf-8'),
                      method='POST')
```

Vous remarquez que notre *payload* va être transformé en chaîne de caractères via la fonction `dumps` de la librairie standard `json`, puis encodé en chaîne de type `utf-8`.

Il ne reste plus qu'à envoyer tout cela !

```
response = request.urlopen(req)
```

Et constater avec effroi que cela n'a pas marché ! 🍊 Horreur et frustration !

En fait, il nous manque deux choses dans les *headers* de notre requête.

Premièrement, il faut préciser que nous envoyons bien du json en rajoutant le header `'Content-Type': 'application/json'`.
Ensuite, les serveurs Discord n'aiment pas trop les requêtes fournies par la librairie python. On va donc se faire passer pour un navigateur web en précisant un *user-agent*, `'user-agent': 'Mozilla/5.0 (X11; U; Linux i686) Gecko/20071127 Firefox/2.0.0.11'`. On met tout cela dans un dictionnaire, puis on rajoute l'option `headers` dans notre requête.

Voici le code complet :

```
#!/usr/bin/env python3

import json
from urllib import request
from urllib.error import HTTPError


WEBHOOK_URL = 'https://discordapp.com/api/webhooks/xxx/yyy'

# La payload
payload = {
```

```
    'content': "Salut les z'agrumes ! Ça zeste ?"
}

# Les paramètres d'en-tête de la requête
headers = {
    'Content-Type': 'application/json',
    'user-agent':
        'Mozilla/5.0 (X11; U; Linux i686) Gecko/20071127 Firefox/2.0.0.11'
}

# Enfin on construit notre requête
req = request.Request(url=WEBHOOK_URL,
                      data=json.dumps(payload).encode('utf-8'),
                      headers=headers,
                      method='POST')

# Puis on l'émet !
try:
    response = request.urlopen(req)
    print(response.status)
    print(response.reason)
    print(response.headers)
except HTTPError as e:
    print('ERROR')
    print(e.reason)
    print(e.hdrs)
    print(e.file.read())
```

*i* Le saviez-vous ?

Discord permet une mise en forme des messages avec une partie de la syntaxe Markdown. Cette mise en forme est aussi supportée dans les messages faits par des bots !

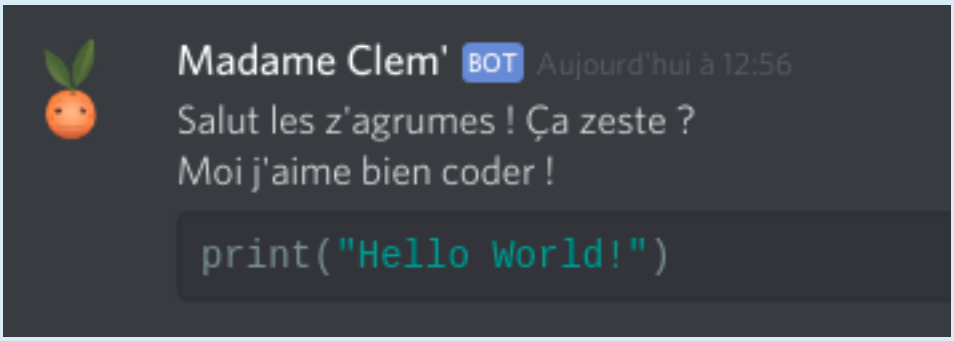


FIGURE 2.5. – Un message tout simple, mais mis en forme !

## 2.2. Envoyer des messages plus « riches »

Les *webhooks* Discord nous permettent aussi de mettre en œuvre des messages plus complexes grâce au format *embed*.

Ce format permet de poster des messages possédant un titre, un corps et même plus encore ! Voyez plutôt :

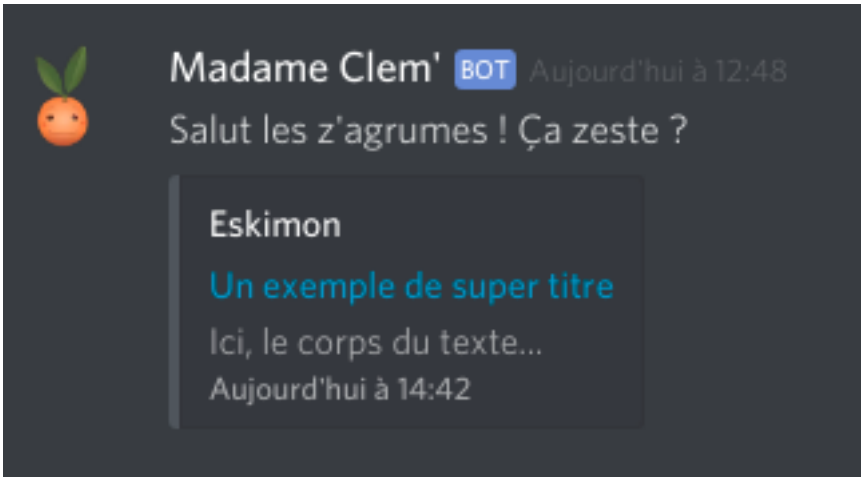


FIGURE 2.6. – Un contenu un peu plus riche

Pour réussir ce tour de force, tout repose sur ce morceau de la documentation . Concrètement, plusieurs champs sont disponibles pour construire un objet et tous sont facultatifs. Afin de ne pas faire une longue liste à la Prévert, voici plutôt un exemple ayant permis l'illustration précédente :

```
payload = {
    'content': "Salut les z'agrumes ! Ça zeste ?",
    'embeds': [
        {
            'title': 'Un exemple de super titre',  # Le titre de la carte
            'description': 'Ici, le corps du texte...',  # Le corps de la carte
            'url': 'https://zestedesavoir.com/on-peut-aussi-mettre-une-url',  # Si vous voulez faire un lien
            'timestamp': '2019-02-28T13:42:00',  # Si vous voulez horodater le contenu
            'author': {'name': 'Eskimon'},  # Pourquoi pas mettre des auteurs ?
        },
    ]
}
```

Mais ce n'est pas tout ! Si vous avez consulté la documentation, vous avez dû voir que l'on peut aussi poster des vidéos, des images, etc. et même changer un peu le style de la «carte». Bref, c'est un format assez versatile qui peut vite s'avérer pratique pour mettre en œuvre des contenus plus riches.

---

Et voilà, notre petit bot peut maintenant être complété pour faire de plus en plus de choses !

Si vous voulez mettre en pratique tout cela, un canal dédié à vos entraînements est disponible sur le Discord non-officiel de Zeste de Savoir . N'hésitez pas à y faire un tour et à coder votre propre solution si vous voulez vous entraîner ou vous amuser (il suffit de demander les clés aux admins sur place ) !

Les *webhooks* ne sont que la partie émergée des possibilités de bots possibles sur Discord. Si vous êtes motivés, plongez-vous dans la doc pour en savoir plus et faire des bots bien plus évolués, capables par exemple de répondre aux messages des utilisateurs. Pour cela, sachez aussi qu'il existe des bibliothèques pour vous simplifier le travail comme discord.py . Amusez-vous bien !

Profitons aussi de cette conclusion pour glisser un petit remerciement à tout ceux qui sont intervenus de près ou de loin pour rendre ce tuto le plus propre possible.