Discord Bot

Mickaël Razzouk
Malo Beauchamps
Matthieu Correia
Arthur Wambst
Othmane Ziyad

EPITA
2024

Table des matières

1 Introduction

1.1 Objectif

La Banquise vous propose ce TP dont le but est de coder un bot discord et de l'héberger sur le serveur mis à votre disposition en vous aidant des outils présentés ci-dessous. Il est fortement recommandé de lire attentivement le sujet, et n'hésitez pas à poser vos questions aux membres de la Banquise !

1.2 Outils

— **Python** est un langage de programmation relativement simple, il est facile à apprendre et à comprendre. Il peut être utilisé dans de nombreux domaines : le développement de site web, l'intelligence artificielle, les jeux vidéos ou encore le prototypage. Une grande communauté se concentre autour de ce language et produit de nombreux tutoriels, dont une grande partie s'adresse aux néophytes.

— **SSH** (*Secure Shell*) est un protocole de communication sécurisé utilisé pour accéder à distance à des ordinateurs et pour exécuter des commandes sur ceux-ci.

— **Vim** est un éditeur de texte puissant, mais compliqué à maîtriser parfaitement (vous y serez contraints lors de l'ING1, alors autant commencer maintenant !). Il s'articule autour de 4 modes principaux : *Normal*, *Insertion*, *Visual* et *Command*, desquels nous aborderons surtout les 2 premiers lors de ce TP.

Pour créer un bot discord, il sera indispensable d'avoir un compte discord. Vous en aurez besoin ainsi que du site. discord developer.

Vim

- Changer de mode : Normal $\underset{\textit{échap}}{\overset{\textit{i/a}}{\rightleftharpoons}}$ Insertion

---

- Naviguer (Normal) :

  $\leftarrow$ gauche : *h* $\quad\quad$ $\uparrow$ haut : *k*

  $\rightarrow$ droite : *l* $\quad\quad$ $\downarrow$ bas : *j*

- Intéractions (Normal) :

  — sauvegarder : *:w*

  — quitter : *:q*

  — sauvegarder et quitter : *:x* ou *:wq*

2 Se connecter en SSH

Pour se connecter en ssh il faut utiliser la commande ssh avec quelques paramètres :

1. le nom d'utilisateur, dans notre cas, ce sera votre login papier
2. le nom de domaine du serveur, dans notre cas, ce sera la-banquise.fr
3. le port sur lequel le serveur ssh écoute, dans notre cas il vous sera donné à chacun.

Ainsi, si votre prenom est Justin et votre nom de famille est Ptipeu et que votre port est le 6667, alors la commande ressemblera à

```
ssh Login@la-banquise.fr -p 6667
```

Un mot de passe vous sera aussi communiqué. Vous en aurez besoin pour vous connecter après avoir rentré cette commande ainsi que dans le reste du tp.

3 Création du bot

3.1 Avant de commencer

Avant de commencer à coder notre bot, il faut lui créer un compte qui lui permettra de se connecter à discord. Pour cela, rendez-vous sur le site discord developer et suivez les étapes suivantes :

Creation

- Connectez-vous à votre compte Discord
- Cliquez sur ***New Application*** pour créer une nouvelle application
- Accédez à l'onglet ***Bot*** dans le menu sur le côté gauche
- Vous pouvez personnaliser les paramètres de votre bot comme son nom d'utilisateur, son image, etc...
- Obtenez le token d'authentification nécessaire pour controller votre bot à partir de Python

(Sur discord developer : Bot -> Token -> RESET TOKEN

- Copiez-le ici et gardez-le précieusement

3.2 Mise en ligne

Pour pouvoir tester votre bot au cours de ce TP, nous allons voir comment l'inviter sur un serveur dans lequel vous êtes admin.

Remarque

Il est fortement recommandé de créer un serveur test dédié.

Inviter le bot

- Retournez sur la page Discord dev, sélectionnez votre appli
- Naviguez jusqu'à l'onglet ***OAuth2*** puis ***URL Generator***
- Sélectionnez uniquement la permission ***bot*** dans les cases à cocher
- Le detail des permissions apparait plus bas. Pour les besoin de ce TP, au minimum ***Read Messages/View Channels*** et ***Send Messages*** sont requis.
- Une fois le lien généré, copiez-le dans votre navigateur, choisissez ensuite votre serveur test
- Verifiez que le bot ait rejoint le serveur. Remarquez qu'il est pour le moment hors-ligne

Remarque

Il faut en général sélectionner le minimum de permissions, de façon à limiter la casse si le token du bot est compromis.

3.3 Développement du bot

3.3.1 Connecter le bot à son compte

Maintenant que nous avons créé notre bot sur Discord, nous allons le connecter à notre code Python en utilisant la bibliothèque *discord-py*. Voici les étapes pour connecter votre bot à son compte :

Connecter le bot

- Créez un fichier Python pour votre bot (ex : `bot.py`) et ouvrez-le : `vim bot.py`
- Importez la bibliothèque `discord-py` dans votre fichier Python en ajoutant la ligne `from discord import app_commands, Intents, Client`
- Creez une variable `TOKEN` egale au token stocke plus tot

pensez a mettre le token entre guillemets pour que python l'interprete comme du texte, ex : TOKEN = "ThisIsMyToken"

- Ajoutez les lignes suivantes :

```
intents = Intents.default()
client = Client(intents=intents)


@client.event
async def on_ready():
    # TODO


client.run(TOKEN)
```

Voici un exemple de ce a quoi votre fichier ressemble, avec quelques commentaires explicatifs :

Connecter le bot - Verifications

```
from discord import app_commands, Intents, Client
# Le token que vous obtenez à la création du bot, il permet au
# bot de se connecter sur son compte, ne le partagez pas!
TOKEN = "discord_token"

# Nous verrons la notion d'intent dans la prochaine partie
intents = Intents.default()
client = Client(intents=intents)

#   Espace pour la suite du TP

# bonus : des informations sur le bot se trouvent dans
# l'objet client; essayez client.user
@client.event
async def on_ready():
    ####### TODO : print() qui affiche un message au lancement
   du bot #######
client.run(TOKEN)
```

Il ne reste plus qu'a remplir la partie TODO

Utiliser la fonction print("messageQueVousVoulez") pour afficher un message, qui vous permettra de savoir qunand le bot sera lance.

Connecter le bot - Fin

- Supprimez le `#TODO` et completez la ligne.

Bonus : des informations sur le bot se trouvent dans l'objet client ; essayez client.user

Maintnant, la base de votre bot est prete ! Sauvegardez le fichier (`ECHAP`, puis `:wq` pour sauvegarder et quitter vim), puis executez-le en tapant `python bot.py` dans la console.

Si tout fonctionne, vous ne devriez voir aucune erreur et voir le message que vous avez mis dans le print() plus tot.

3.3.2 Intents Discord

Les *intents* Discord sont une fonctionnalité introduite par l'API Discord pour offrir un contrôle plus précis sur les événements que votre bot reçoit. Ils permettent aux développeurs de spécifier les types d'événements que leur bot recevra.

Nous allons maintenant faire repondre votre bot a certains messages. Cela représente beaucoup de données pour un bot qui est présent sur des milliers de serveurs et peut potentiellement causer des problèmes de vie privée.
C'est pour cela que les développeurs de bots doivent déclarer explicitement les *intents* que leurs bots utiliseront. Cela aide Discord à gérer la charge sur ses serveurs et garantit que les bots ne reçoivent que les données nécessaires.

Intents

- Dans notre cas, c'est l'*intent* message_content qui sera utilisé, vous devez donc rajouter la ligne `intents.message_content = True` après la déclaration de la variable `intent` afin de recuperer chaque message avec `message.content`
- Il vous faut aussi activer l'intent sur le panel de votre bot (`Bot -> Message Content Intent`) sur le site discord developer.

Lors de l'implémentation d'un bot Discord, il est essentiel de bien réfléchir aux *intents* dont votre bot a besoin selon sa fonctionnalité. Les *intents* inutilisés doivent être désactivés pour minimiser la charge sur les serveurs Discord et améliorer les performances. Il existe bien d'autre *events* qui permettent de rendre les bots bien plus utiles, vous en trouverez une liste exhaustive sur la documentation Discord.

3.3.3 Ajouter des events

Maintenant que notre bot est connecté à Discord, nous pouvons lui ajouter des *events* pour qu'il puisse réagir à des événements.

Voici comment ajouter un simple événement *on_message* qui répondra à un message «ping» avec «pong» :

Intents

- Vous pouvez ajouter ce code à votre fichier Python **au-dessus** de la fonction `on_ready()`.

```
@client.event
async def on_message(message):
    if message.content == 'ping':
        await message.channel.send('pong')
```

Penser ensuite a sauvegarder puis quitter le fichier (`ECHAP` puis `:wq`) et pensez a **redémarrer** (`python bot.py`) votre bot.

3.3.4 Ajouter des commandes

À une époque, les bots Discord étaient configurés pour surveiller tous les messages et ne réagir que s'ils détectaient un caractère spécial au début du message, par exemple " ?". C'était la méthode la plus courante pour détecter une commande et effectuer une action en réponse, comme dans les exemples suivants :

— `?help` : pour obtenir une liste de commandes.

— `?mute <user>` : pour interdire à un utilisateur l'envoi de messages.

— `?coinflip` : pour lancer une pièce.

Cependant, cette méthode présentait un inconvénient majeur : elle nécessitait que l'*intent* de message soit activé en permanence. C'est pourquoi Discord a introduit nativement la notion de commandes sur leur plateforme, ce que nous allons apprendre à utiliser dans cette section.

Avant tout, il vous faudra ajouter l'import suivant : `from discord import app_commands`

Il faudra ensuite déclarer le *command tree* au même endroit que vos *intents* comme cela : `tree = app_commands.CommandTree(client)`

Votre code devrait donc ressembler à ceci :

```
from discord import app_commands, Intents, Client

TOKEN = <discord_token>
intents = Intents.default()
client = Client(intents = intents)
intents.message_content = True
tree = app_commands.CommandTree(client)

@client.event
async def on_message(message):
    print(message.content)
    if message.content == 'ping':
        await message.channel.send('pong')

@client.event
async def on_ready():
    print(str(client.user) + " : message")
client.run(TOKEN)
```

Vous pouvez maintenant créer de nouvelles commandes, pour cela il vous faut déclarer une commande et la fonction associée :

```
@tree.command(name = "plus", description = "simple addition")
async def addition_cmd(ctx,a:int,b:int):
    await ctx.response.send_message(f"The calculation result
```

```
    is:\n{a+b}" )
```

Après avoir créé votre première commande, il faut informer discord de son existence en synchronisant votre tree. Vous allez utiliser la fonction : `await tree.sync()`.

Petit conseil : appelez-la à un endroit exécuté à chaque démarrage de votre bot !

4 Sauvegarde et Gestion de Versions

Après avoir terminé un travail, il est toujours judicieux de le sauvegarder afin de pouvoir y accéder ultérieurement pour des modifications futures ou pour reprendre le travail depuis un autre ordinateur. Pour cette raison, plusieurs techniques sont disponibles, allant des plus simples aux plus complexes.

4.1 Google Drive

Des services de stockage en ligne tels que Google Drive offrent la possibilité de conserver différents fichiers sans nécessiter gestion de

version. Cela reste une solution simple, mais extrêmement limitée.

4.2 Git et GitHub

Git est une technologie particulièrement intéressante et largement utilisée dans le domaine de la programmation et de l'informatique en général. Vous avez peut-être déjà entendu parler de GitHub, une plateforme populaire basée sur Git.

4.2.1 Inscription sur GitHub

Nous vous recommandons de créer un compte sur GitHub pour sauvegarder votre travail. Pour ce faire, visitez le site de GitHub et cliquez sur "S'inscrire" en haut à droite. Suivez les instructions du formulaire d'inscription et vérifiez votre adresse e-mail.

4.2.2 Création d'une Clé SSH

Une clé SSH permet une authentification sécurisée à chaque opération effectuée sur GitHub. Pour en créer une, exécutez la commande suivante dans votre terminal :

```
ssh-keygen -t ed25519
```

Une fois la clé créée, vous trouverez deux nouveaux fichiers dans le dossier .ssh (un dossier caché) : id_ed25519 et id_ed25519.pub. Assurez-vous de ne jamais partager le contenu du fichier id_ed25519. Copiez le contenu de id_ed25519.pub en utilisant la commande :

```
cat ~/.ssh/id_ed25519.pub
```

Collez ensuite le contenu de la clé publique dans les paramètres de votre compte GitHub (Photo de profil → Settings → SSH and GPG keys → New SSH key). N'oubliez pas de lui donner un nom et une description.

Votre clé SSH est maintenant configurée et prête à être utilisée en toute sécurité.

4.2.3 Envoi du Code sur GitHub

Vous pouvez maintenant passer à l'étape de sauvegarde effective de votre travail sur GitHub :

1. Créez un nouveau "repository" en cliquant sur le bouton "+" en haut à droite, puis sur "New repository".
2. Entrez le nom, la description et déterminez s'il doit être privé

ou non, puis cliquez sur "Create Repository".

3. Dans un terminal, naviguez jusqu'au dossier de votre projet et exécutez les commandes suivantes :

Attention !

Assurez-vous de modifier certains champs selon votre configuration, ne copiez pas ces commandes directement.

```
git init
git add .
git commit -m "Changez cette ligne pour décrire les modifications effectuées"
git branch -M main
git remote add origin git@github.com:[votre_nom_d_utilisateur]/[nom_de_votre_repository].git
git push -u origin main
```

Git est un sujet très vaste, et vous pouvez trouver de nombreux guides en ligne sur des plateformes telles que GitHub, apprendre de manière interactive sur des sites comme GitBook, ou encore lire le très complet Git Book.

5 Conclusion

À travers ce TP vous avez vu comment créer, connecter, mettre en ligne un bot, le faire réagir à des événements, et enfin comment lui ajouter des commandes. Vous pouvez maintenant explorer les possibilités que vous offre le module discord de Python ! Si vous avez des questions, posez-les aux membres de La Banquise !