Deux « bots » de plus pour le fédivers

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 7 avril 2019

https://www.bortzmeyer.org/deux-bots-fediverse.html

Je viens de mettre en service deux « bots », sur le réseau social « fédivers ». Ce court article est là pour documenter les techniques utilisées, si vous voulez vous-même réaliser un tel « agent logiciel autonome » sur ce réseau.

Un « bot », dans le contexte des réseaux sociaux, est simplement un logiciel qui, sans intervention humaine, va écrire sur le réseau social, parfois en réponse à des demandes des humains. C'est un concept très ancien (il y en avait déjà sur IRC) mais je vais me focaliser ici sur des bots participant au **fédivers** (également écrit fedivers, "*fediverse*", féediverse, etc). J'avais déjà documenté en anglais un bot qui répondait à des requêtes DNS <https://www.bortzmeyer.org/fediverse-bot.html>, les deux bots décrits ici en français sont plus simples, ils n'écoutent pas les messages, ils ne font qu'écrire. Cela permet de ne pas avoir un démon tournant en permanence (et qu'il faut superviser, redémarrer, etc) mais juste de les lancer de temps en temps depuis cron : ils écrivent puis retournent se coucher.

Ces deux bots sont `balladependus`, qui écrit le texte du poème « La ballade des pendus », de François Villon, et `voirapp`, qui récite la chanson « Voir » de Jacques Brel. À quoi ça sert d'envoyer des poèmes sur le fédivers ? À rien, mais plusieurs bots le font déjà, comme `LInternationale` qui nous réveille avec les paroles de l'Internationale (une version préliminaire, pas les paroles les plus connues).

Les deux bots sont hébergés par l'**instance** fédivers `botsin.space`, spécialisée dans les bots.

Le premier, `balladependus`, est écrit en shell, le second, `voirapp`, en Python. Voyons d'abord le premier.

Le code source de `balladependus` est distribué ici (en ligne sur `https://www.bortzmeyer.org/files/ballade-pendus.sh`). Il repose sur l'excellent programme madonctl <https://github.com/McKael/madonctl>, qui permet d'accéder au fédivers depuis la ligne de commande. madonctl utilise l'API du serveur Mastodon (`botsin.space` est un Mastodon). Normalement, Pleroma <https://pleroma.social/> gère également cette API et donc madonctl devrait marcher avec Pleroma, mais je n'ai jamais essayé. Autrement, que fait le programme ? En commençant du début :

— Il récupère sur la ligne de commande ses deux arguments, le fichier contenant le poème, et le nom du fichier de configuration (le même programme peut être utilisé pour plusieurs poèmes et plusieurs comptes).
— Il lit un fichier de configuration, contenant les paramètres spécifiques à un compte fédivers, notamment le jeton d'autorisation ; ce fichier a été créé avec `madonctl config dump -i YOURINSTANCE -L YOURID -P YOURPASSWORD > /.config/madonctl/YOURCONFIG.yml`. Il faut donc s'être créé un compte sur l'instance préalablement (en n'oubliant pas de cocher, dans le profil, la case « Je suis un bot »).
— Il lit ensuite un fichier d'état (`$checkpoint`) qui indique à quelle strophe du poème on en est. Le logiciel ne tourne pas en permanence et c'est donc ce fichier d'état qui lui permettra de ne pas partir de zéro à chaque fois. Le même fichier contient un identificateur du pouète (message sur le fédivers) précédemment envoyé, de façon à organiser toutes les strophes du poème en un seul fil.
— Ensuite, il lit le fichier contenant le poème, une ligne vide indique le passage au pouète suivant. (Le fichier a été créé manuellement avec un éditeur.)
— Le bot attend ensuite un nombre aléatoire de secondes, pour mettre un peu de variété dans les messages. Notez qu'il n'existe pas de moyen standard en shell de faire des nombres aléatoires <https://www.bortzmeyer.org/unsort.html>.
— Enfin, on écrit le pouète avec madonctl, qui prend en argument le fichier de configuration de ce compte (paramètre `--config`) et, sauf pour la première strophe, l'identificateur du pouète précédent (paramètre `--in-reply-to`). On note en retour l'identificateur (`Status ID`) du pouète.
— Enfin, il n'y a plus qu'à écrire identificateur du pouète et numéro de la strophe dans le fichier d'état.

Le programme est lancé automatiquement par cron, envoyant une strophe toutes les trois heures :

```
# Ballade des pendus
35 0,3,6,9,12,15,18,21 * * * ballade-pendus.sh ballade-pendus.txt ballade
```

Et l'autre programme, celui derrière `voirapp` ? Il est écrit en Python, pour le plaisir de la variété. Il dépend de la bibliothèque Mastodon.py <https://github.com/halcy/Mastodon.py>. Il est plus court que la version shell, en partie parce que Python offre des possibilités supérieures, mais également parce qu'il lui manque certaines fonctions. Par exemple, il est moins générique, le nom du poème est en dur dans le code. Le source est également disponible (en ligne sur `https://www.bortzmeyer.org/files/voir.py`).

Comment fonctionne ce programme ? Il ressemble beaucoup au précédent. Il ouvre le fichier d'état (`CHECKPOINT`), y lit l'identificateur du pouète précédent et le numéro de la strophe. Il lit le fichier contenant la chanson (même formatage que pour le script en shell), et attend une durée aléatoire avant d'envoyer, et enfin écrit le nouveau fichier d'état. Pour envoyer, il se connecte à une instance Mastodon (`mastodon = Mastodon(...)`) et envoie le pouète via `mastodon.status_post(...)`. Avant cela, il aura fallu s'enregistrer auprès de l'instance, pour fabriquer les fichiers contenant les lettres de créance (`voirapp_clientcred.secret` et `voirapp_usercred.secret`). Cet enregistrement peut également se faire en Python (une fois suffit, les lettres de créance seront enregistrées) :

```
Mastodon.create_app(
     'voirapp',
     api_base_url = 'https://botsin.space/',
     to_file = 'voirapp_clientcred.secret'
)
mastodon = Mastodon(
    client_id = 'voirapp_clientcred.secret',
    api_base_url = 'https://botsin.space/'
)
mastodon.log_in(
    USER,
    PASSWORD,
    to_file = 'voirapp_usercred.secret'
)
```

Le programme est ensuite lancé depuis cron, comme le précédent.

https://www.bortzmeyer.org/deux-bots-fediverse.html