Auteurs: Habib Belaribi, Marc Cenon, Pierre Chevalier, Guzel Katnik, Cyril Millet

Date: 27/03/2021

À destination de:
Cyril Roelandt & Samuel Thibault

Projet tuteuré
-
Rapport technique

Synthèse

Dans le cadre du projet tuteuré de la licence professionnelle ADSILLH, notre groupe a choisi de contribuer au logiciel libre Searx.

Il s'agit d'un méta-moteur de recherche permettant de ramener des résultats à partir de moteurs de recherche généralistes, mais aussi à partir de divers sites internet, notamment grâce à des APIs.

Searx se veut particulièrement respectueux des données à caractère personnel des utilisateurs. Il comporte à ce jour plus de 70 sources de données.

La majeure partie du logiciel est codée en Python, il est basé sur le micro-framework Flask. Le code est hébergé sur github: https://github.com/searx/searx.

Nos contributions ont porté sur les thèmes suivants:

- documentation;
- interface utilisateur: corrections de bogue et traductions;
- sources de données : réparations et ajouts de sources de données.

L'accueil par la communauté des développeurs fut très chaleureux, et nos contributions ont bénéficié d'attentions et de corrections constructives par deux des principaux développeurs.

Par ailleurs, nous avons fait des installations de Searx sur des serveurs opérationnels, sur diverses configurations: Docker, Kubernetes, Yunohost, Debian.

Remerciements

Nous tenons à remercier Cyril Roelandt pour sa patience et sa disponibilité, l'association Aquilenet pour son accueil et pour la mise à disposition d'un serveur virtuel, FX Lesaffre pour avoir veillé sur quelques *pull requests*, l'ensemble des contributeurs de Searx qui nous ont épaulés durant nos travaux, et enfin toutes et tous les inconnus ayant pris le temps de documenter sur leurs blogs et autres forums.

Table des matières

1.Introduction

1.1.Choix du projet

Notre choix s'est porté sur le méta-moteur de recherche Searx, en raison du caractère dynamique du projet, du principal langage utilisé (Python), du caractère open-source du logiciel qui est sous licence libre GNU GPL AFFERO V3.

1.2.Fonctionnalités

Searx a pour caractéristique et pour promesse la non divulgation des données à caractère personnel des utilisateurs. Il permet de ramener des résultats à partir des moteurs de recherche généralistes grand public, mais aussi depuis des sites internet proposant des collections de contenus, notamment grâce aux APIs fournies par les dits sites.

Une autre fonctionnalité de Searx, à l'inverse d'un moteur de recherche "généraliste", est de pouvoir faire des recherches thématiques fines dans diverses disciplines (ex : sciences, informatique…).

Searx est inspiré du projet Seeks, moteur de recherche disponible sous la même licence que Searx. Il se présente, entre autres, sous la forme d'un proxy web qui intercepte les requêtes faites à d'autres moteurs de recherche, les soumet aux moteurs activés dans la configuration, recueille et recoupe les résultats pour ensuite les présenter à l'utilisateur final.

Searx peut être utilisé au travers d'une instance publique mais il peut aussi être installé localement, dans l'optique d'un auto-hébergement, pour disposer de sa propre instance (ce qui rend la fonction de non divulgation des données personnelles moins pertinente, si l'on se réfère à l'architecture technique du projet dans le paragraphe suivant 1.3). De nombreuses instances publiques sont disponibles, la liste est consultable à l'adresse suivante : https://searx.space/

Il est assez léger pour être installé sur un ordinateur du type Raspberry Pi.

1.3.Fonctionnement

Searx respecte la vie privée de ses utilisateurs dans la mesure où:

- il ne partage pas les adresses IP de ses utilisateurs avec le site de recherche qu'il questionne pour les requêtes;
- les cookies, qui peuvent être utilisés pour des formes de traçage des internautes, sont bloqués afin d'éviter tout type de recherche personnalisée suivant le profil de l'utilisateur;
- de plus, Searx ne partage aucune donnée avec des tiers;
- par défaut, les requêtes sont effectuées en utilisant des requêtes de type POST. Par conséquent, elles ne figurent ni dans les journaux du système, ni dans l'historique des URLs consultées.

Searx agrège à ce jour plus de 70 sources de données.

Il comporte plusieurs rubriques/catégories:

- général,
- fichier,
- image,
- informatique (forges git, torrent, news spécialisées, etc.),
- cartographie,
- sciences,
- réseaux sociaux,
- vidéos.

Voici quelques exemples de sources de données supportées par Searx: duckduckgo, wikipedia, bing, deezer, flickr, gitlab, github, google, kickasstorrent, Nyaatorrent, PirateBay, qwant, stackoverflow, twitter, yahoo, peertube, photon, etc.

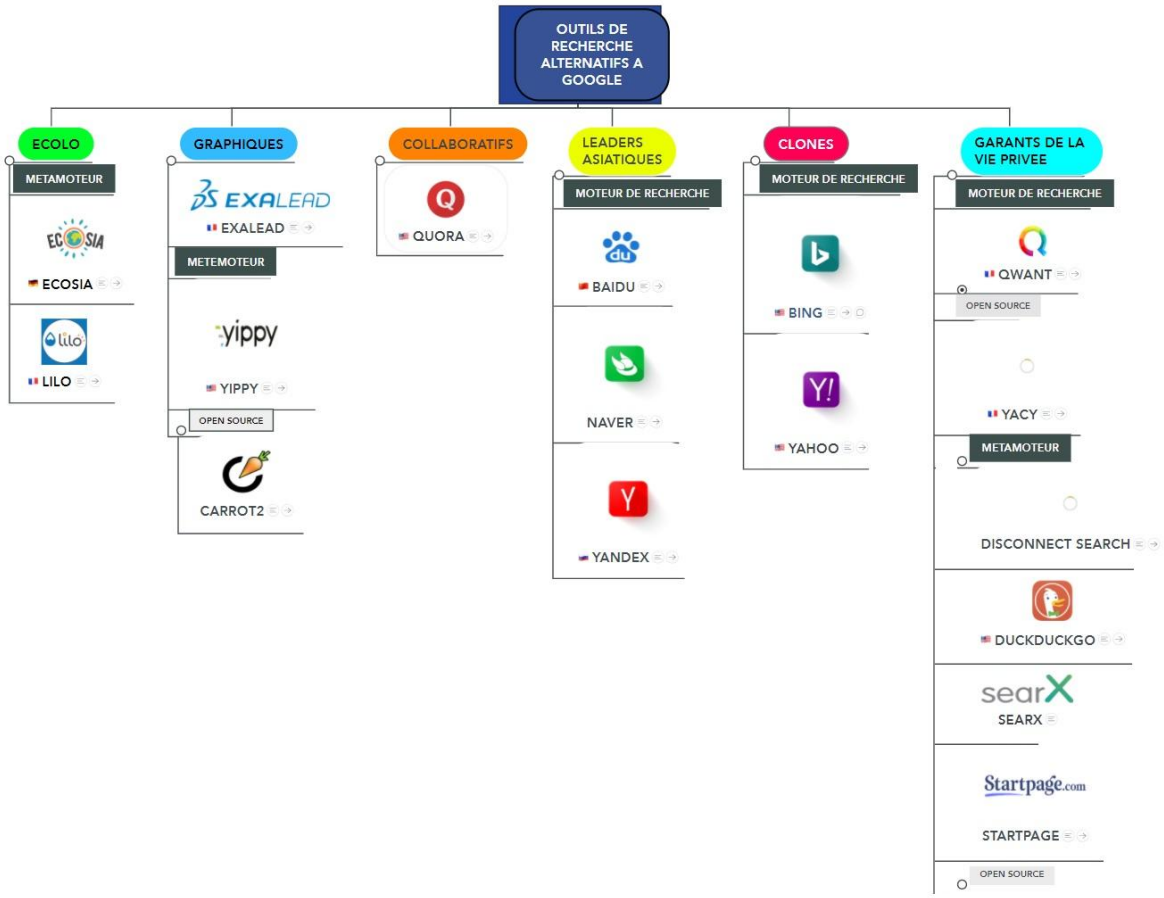


Fig. 1: Grande variété dans les sources de données recherchées: ceci illustre qu'il existe bien des alternatives, en matière de recherches sur Internet, aux moteurs de recherche conventionnels, comme Google

1.4. Quelques indicateurs concernant le projet

- Création du projet en 2014;
- Un peu plus de 4000 commits;
- Une vingtaine de contributeurs, dont 7 contributeurs très actifs (+ de 100 commits chacun) et 7 contributeurs assez actifs (10 à 25 commits chacun) depuis l'initiation du projet;
- Projet financé par la fondation NLnet (https://nlnet.nl/).

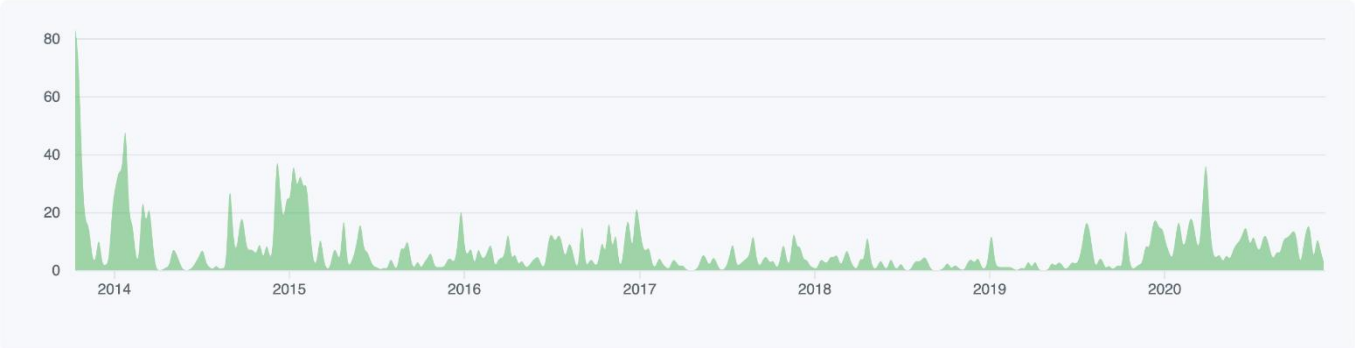


Fig. 2: Historique de l'activité du projet, en quantité de contributions à la branche « master » (« merge commits » exclus)

Le graphique suivant, représentant le nombre de commits sur les quatre derniers mois, permet de constater que le projet est vivant et plutôt actif.

Fig. 3: Nombres de commits journaliers au cours des quatre derniers mois

Le graphique suivant permet de constater que sur l'année 2020, il y a eu beaucoup de modifications du code par rapport aux autres années, ce qui montre l'intérêt actuel porté à ce projet.

Fig. 4: Graphique montrant les ajouts et retraits de code de 2014 à 2020

Searx n'est pas un projet de grande envergure tel que VLC ou Firefox. Il est néanmoins actif et dynamique, avec une communauté volontaire, ce qui est en phase avec la qualité de l'accueil que nous y avons reçu.

2.Équipe, communauté

Adam Tauber, alias ASCIIMOO, est le créateur du projet. Il est extrêmement discret sur le net (pas de photographie, ni d'interview personnelle ou autre).

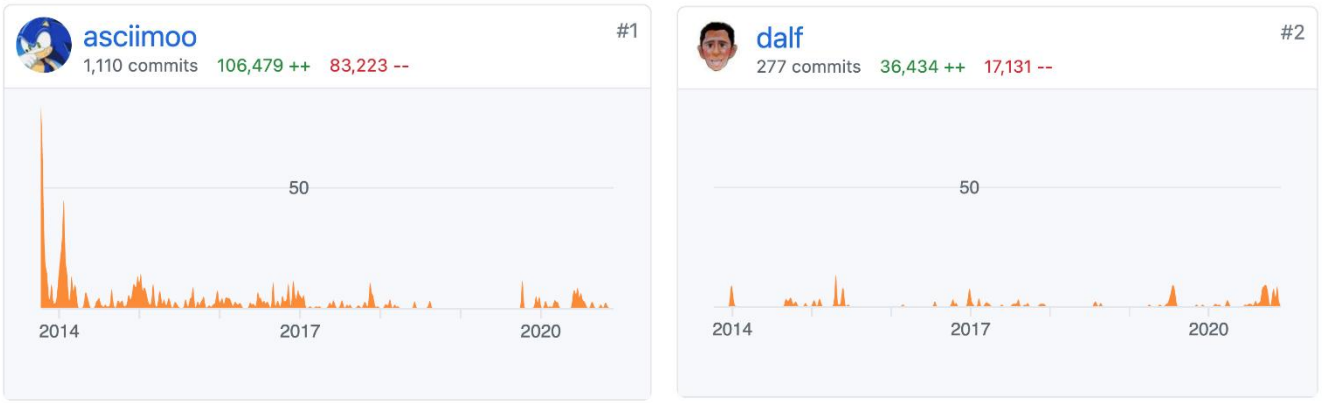


Fig. 5: Deux contributeurs importants, dont l'initiateur du projet

@dalf, @return42, @kvch sont les contributeurs qui ont incorporé et validé la plupart de nos *pull requests*.

La communauté de développeurs de Searx échange :

- par le biais de github (https://github.com/searx/searx), mode de communication le plus intensément utilisé ;
- via un canal IRC (@freenode#searx) que nous avons visité.

D'autres utilisateurs et contributeurs échangent par d'autres canaux, par des listes de diffusions privées, des forums, des blogs, qui documentent ou partagent des informations sur Searx (ex : https://framacloud.org/fr/cultiver-son-jardin/searx.html).

3.Structure de Searx

3.1.Le micro-framework Flask

Flask[1] est un micro-framework de développement web en Python. Flask utilise le moteur de template Jinja2 et la boîte à outils WSGI de Werkzeug[2].

La structure de la webapp Flask appliquée à Searx est globalement la suivante :

- à la racine le fichier *webapp.py* : fichier python principal sous Flask qui définit les fonctions fondamentales (routes, templates, fichiers statiques et opérations à servir)
- autres <*fichiers*>.*py*
  - dans le sous-répertoire */searx/engines* notamment : contient entre autres les fonctions qui vont requêter les diverses sources de données selon les règles de leur API
- à la racine le répertoire */templates* : contient tous les <*fichiers*>.*html* qui définissent la structure d'affichage des pages servies par l'application (incluant également la syntaxe jinja, pour la définition des macros par exemple)
- à la racine le répertoire */static* : contient tous les fichiers de forme (styles, thèmes) reliés aux templates html.

3.2.Architecture pour une instance publique

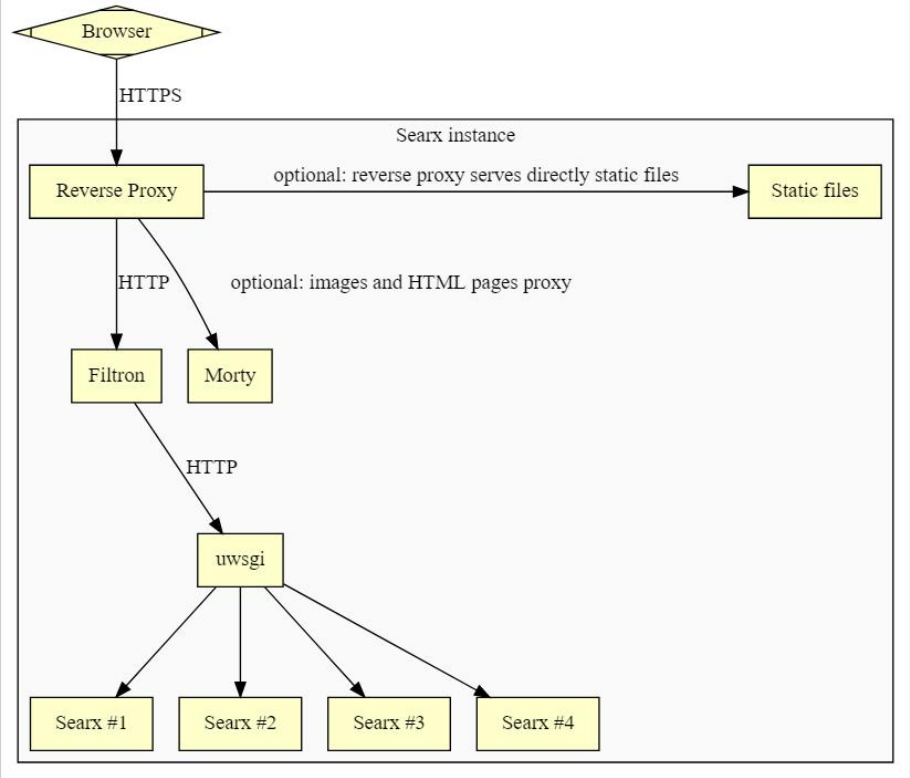


Fig. 6: Architecture, fonctionnement des briques logicielles pour une instance publique de Searx

1 https://flask.palletsprojects.com/en/1.1.x/
2 https://palletsprojects.com/p/werkzeug/

- **Reverse Proxy :** Apache ou Nginx par exemple, se positionne entre la webapp uWSGI et internet.
- **Morty** (en référence à Rick & Morty) est un proxy de résultats pour Searx. Il va réécrire les pages HTML en enlevant les tags et attributs potentiellement malicieux et permet également d'éviter la fuite d'information vers des tiers.
- **Filtron** est un reverse proxy http qui va filtrer les requêtes suivant un ensemble de règles définies.

Ces deux plugins ne sont pas requis pour le fonctionnement de base de Searx, mais ils sont recommandés par les mainteneurs pour la création et l'administration d'une instance publique.

uwsgi est le protocole utilisé pour communiquer avec les autres serveurs, sur laquelle se base le logiciel uWSGI. Ce dernier est utilisé pour servir la partie python (Flask) avec les serveurs qui utilisent le protocole uwsgi.

4.Organisation du travail

4.1.Dépôts git

Chacun de nous a interagi avec son propre compte sur github. Nous avons échangé entre nous avant de soumettre chaque Pull Request.

Liste de nos dépôts individuels :

- Cyril : https://github.com/mrwormo/searx
- Guzel : https://github.com/GazoilKerozen/searx
- Habib : https://github.com/habsinn/searx
- Marc : https://github.com/spongebob33/searx
- Pierre : https://github.com/pierrechtux/searx et https://github.com/pchg/searx

4.2.Outils utilisés

Le Makefile[3] situé à la racine du projet Searx a permis de procéder à nombre d'opérations de tests et de débogage.

a)Édition du code

Nous avons simplement eu recours à nos éditeurs de texte préférés, avec nos configurations personnalisées. Des outils comme CODE – OSS ou vimdiff se sont avérés très pratiques pour le versionnement git et la comparaison des lignes de code des fichiers modifiés.

b)Débogage et tests

Le logiciel Make[4] s'est avéré particulièrement efficace lors du développement et du débogage. De sa panoplie de commandes nous avons apprécié notamment:

- "make run" démarre, via le Makefile, une instance de Searx pour débogage, et lance un navigateur sur l'instance Searx. Dès lors, à chaque modification de code source, le serveur est automatiquement mis à jour et les changements sont propagés jusqu'au navigateur, dont il suffit de rafraîchir la page;
- "make test" est une autre commande make fréquemment utilisée. Une fois des modifications faites, cette commande permet de lancer toute une batterie de tests.

  Il y a toutefois des fois où nous n'avons pas apporté assez d'attention à certains messages, notamment concernant les spécifications de PEP8 concernant les espaces, des nombres de lignes vides, les indentations, etc. Pour y remédier nous avons utilisé le formateur de python Black[5] qui corrigeait nos fichiers .py avant de les proposer en commit.

Pour ce qui concerne l'ajout de sources de données dans Searx, il a fallu examiner les flux de données renvoyés par les fournisseurs de données: on a eu recours à `curl` ainsi qu'à des outils comme `json_pp` afin de rendre plus lisible les documents JSON renvoyés.

3 https://searx.github.io/searx/dev/makefile.html#makefile
4 https://fr.wikipedia.org/wiki/Make
5 https://pypi.org/project/black/

c)Outils de travail collaboratif

Nous avons eu largement recours à **Mattermost**, alternative open-source à Slack, **Git** bien évidemment. Nous avons également monté une instance **DokuWiki** afin de documenter au fur et à mesure des participations, nous avons utilisé des **framapad** et des **hackmd** pour co-écrire à la volée, nous avons eu recours à des réunions physiques (nous remercions au passage l'association Aquilenet pour son accueil), **NextCloud** et **CollaboraOnLine** (pour le rapport final), ainsi que de nombreuses sessions sur **Big Blue Button** (pour discuter et travailler avec le partage d'écran de nos terminaux respectifs).

5.Nos contributions

Nous avons commencé par les contributions qui nous semblaient les plus aisées, pour graduellement monter en complexité.

Les Pull Requests de nos participations se résument ainsi:

- documentation (PR Pierre)
- interface utilisateur:
  - gruntfile / source map (PR Cyril)
  - reset button - front end (PR Guzel)
  - traductions (PR Guzel)
- sources de données :
  - réparations:
    - moteur invidious engine (PR Cyril)
  - ajout de nouvelles sources:
    - CC engine (PR Cyril),
    - Library of Congress (PR Cyril),
    - Freesound (PR Guzel),
    - Artic (musée de Chicago) (PR Habib),
    - Springer Nature (PR Pierre en cours),
    - Core.ac.uk (PR Marc en cours)

Nous allons d'abord décrire succinctement ces participations, avec des liens vers les *pull requests* et un résumé du travail effectué pour chacune de ces *pull requests*. Les détails de certaines de ces contributions sont développés dans les annexes.

5.1.Documentation

Liste des pull requests :

https://github.com/searx/searx/pull/2273

https://github.com/searx/searx/pull/2276

https://github.com/searx/searx/pull/2277

https://github.com/searx/searx/pull/2301

Lors de la toute première prise de contact avec le logiciel Searx, en suivant les instructions d'installation de Searx, sur cette page https://searx.github.io/searx/admin/installation-searx.html#installation-basic, des erreurs, assez mineures, ont été détectées dans la documentation.

Voilà qui faisait une occasion parfaite de faire une toute première participation qui promettait d'être fort aisée. La suite des opérations infirma assez vite cette dernière assertion.

Il s'agissait de quelques corrections à faire: un mkdir qui manquait, et une faute de frappe à xgd-open au lieu de xdg-open. Il y a aussi une commande sed qui était sans effet, mais c'est un détail non bloquant.

La reconstruction de la documentation s'est avérée incomplète sur notre machine de développement, sans qu'on ait pu en déterminer la raison. En conséquence, la *pull request* a été envoyée «à l'aveugle», sans savoir si la documentation allait être correctement régénérée.

Une autre participation, assez mineure, était la correction d'une faute de frappe: «virualenv» à la place de «virtualenv».

5.2. Gruntfile / source map

https://github.com/searx/searx/pull/2308

L'ajout de l'option "sourceMap dans le fichier de configuration de Grunt permet au logiciel débogueur de rendre lisible la source du fichier Javascript minifié.

5.3. Reset button - front end

https://github.com/searx/searx/pull/2306

Cette *pull request* correspond à l'issue #2009 ouverte en Juin 2020. Il s'agissait d'une erreur à première vue simple à corriger: le bouton d'effacement de la barre de recherche ne fonctionnait pas dans le thème Oscar (thème par défaut).

Après quelques comparaisons avec le thème Simple et des tests de remplacement de valeurs d'Oscar par ceux de Simple (au cas où elles soient définies quelque part en commun)... Finalement, `grunt` (grunt-cli) fut installé pour tester du JS en local.

Sur Simple le bouton était de `type = "button"`. Par conséquent, dans le html d'Oscar une id a été créée pour le champ `#clear_search` et le type a été modifié de "`reset`" à "`button`". Les modifications ont été enregistrées dans `searx/static/themes/oscar/js/searx_src/element_modifiers.js`

La fonction `onclick` a été ajoutée pour se déclencher sur l'élément `#clear_search`. À L'`id` du champ de recherche (`q`) et à la fonction `js document.getElementById(#q)` a été attribuée une chaîne de caractères vide:

```
$("#clear_search").click(function () {
document.getElementById("q").value = "";
});
```

Pour tester, la fonction devait renvoyer "AAAAAAAAAAAAA" dans la console du navigateur. Malheureusement `grunt` l'a pris en compte lors de la *pull request*. Une autre modification du code s'est retrouvée par inadvertance dans la *pull request* (PR), la correction a été aussitôt faite avec `git commit –amend`.

La PR a été d'abord validée par @return42. Quelques semaines plus tard, @dalf a indiqué l'issue #2034 en cours à ce moment-là, qui retardait sa réponse. Dans le cadre de la résolution de ce ticket, les commits ayant trait aux thèmes ont été suspendus et n'ont pas été traités. Nous nous sommes signalés comme un groupe d'étudiants, suite à quoi la PR a été tout de même traitée... après quelques corrections gentiment soufflées par @dalf :

- rebaser la PR et faire `run make theme.oscar` pour intégrer les modifications du thème survenues depuis ma PR sur la branche master;
- il a ensuite demandé de remplacer la classe `active_if_nojs` par `hide_if_nojs` mais le problème venant du fait que le JS est désactivé dans les paramètres du navigateur, on a fini par remplacer dans html : `<button type="button" id="clear_search" class="btn btn-default hide_if_nojs" aria-label="clear_search">icon_remove</button>`, ce qui a pour effet de cacher le bouton si les utilisateurs n'activent pas le JS dans le leurs navigateurs.

Les travaux en cours sur l'issue #2034 ont été également la raison pour laquelle cette PR comportait aussi des fichiers minifiés et autres générés par `grunt`. En résumé, cette PR est la preuve que les apparences peuvent être trompeuses.

5.4. Traduction

Une traduction “au kilomètre” à été réalisée de l’anglais au polonais. Pour faire une traduction ou correction, il faut créer un compte sur https://transifex.com, ajouter des langues dans son profil puis rechercher des projets auxquels participer, et enfin se joindre à l'équipe Searx pour traduire.

De là, il est possible de télécharger un fichier avec les mots à traduire en local ou les traduire un par un dans l'interface.

Une fois la traduction terminée, elle est en attente de révision par d'autres membres. Ainsi, il est possible de réviser les mots déjà traduits ou commenter les erreurs de traduction qui ne collent pas à la syntaxe ou dans un contexte donné. Il s’agit d’une contribution en-dehors du circuit Git, néanmoins intéressante à découvrir.

5.5. Moteur invidious engine

https://github.com/searx/searx/pull/2348

Cette modification du fichier settings.yml désactive par défaut la fonctionnalité de recherche sur https://invidio.us qui n’était plus accessible depuis plusieurs mois.

https://github.com/searx/searx/pull/2451

Depuis l’arrêt de l’instance principale https://invidio.us, la seule façon d’interroger une instance Invidious était de modifier la variable “base_url” pour lui affecter une URL.

J’ai donc proposé la modification afin que Searx choisisse une instance à interroger au hasard à chaque recherche, dans une liste d’instance déclarée dans le fichier “settings.yml”.

Cette modification a permis de clôturer le bug #2161.

5.6. Nouvelles sources

Afin d’ajouter de nouvelles sources de données (ci-après appelées *engines*, tel qu’utilisé dans le code source de Searx) à rechercher dans Searx, il faut:

- ajouter un fichier dans le répertoire searx/engines
- modifier le fichier settings.yml

Fichier <nouvel-engine>.py

Comme il n’existe pas d’API générale qui s’adapterait à chaque source de recherche, il est nécessaire d’implémenter des “adaptateurs”, nommés “engines” dans le projet, et de les positionner dans le répertoire ***searx/engines***.

Certains arguments sont à déclarer dans le fichier ***<nouvel-engine>.py***, alors que d’autres sont à déclarer dans le fichier de configuration ***settings.yml***[6]**.**

Un fichier <nouvel-engine>.py est toujours formaté sur le même modèle, en 3 parties minimum :

1. **La partie déclarative**, dans laquelle nous retrouvons la déclaration des bibliothèques nécessaires, la partie “about”, les variables;
2. **La fonction de recherche *“request”*** qui « forge » et formate l’url dont Searx a besoin pour effectuer la recherche;
3. **La fonction d’analyse des résultats *“response”***, qui va fournir les résultats à Searx pour les afficher sur la page de résultats.

6 https://searx.github.io/searx/dev/engine_overview.html

Exemple avec le fichier ccengine.py:

- Partie déclarative:

```
# SPDX-License-Identifier: AGPL-3.0-or-later
"""

 Creative Commons search engine (Images)

"""

from json import loads
from urllib.parse import urlencode


about = {
    "website": 'https://search.creativecommons.org/',
    "wikidata_id": None,
    "official_api_documentation": 'https://api.creativecommons.engineering/v1/',
    "use_official_api": True,
    "require_api_key": False,
    "results": 'JSON',
}

categories = ['images']

paging = True
nb_per_page = 20

base_url = 'https://api.creativecommons.engineering/v1/images?'
search_string = '&page={page}&page_size={nb_per_page}&format=json&{query}'
```

- Fonction “request” qui sert à construire l’URL de recherche :

```
def request(query, params):

    search_path = search_string.format(
        query=urlencode({'q': query}),
        nb_per_page=nb_per_page,
        page=params['pageno'])

    params['url'] = base_url + search_path

    return params
```

- Fonction qui parse les résultats :

```
def response(resp):
    results = []

    json_data = loads(resp.text)

    for result in json_data['results']:
        results.append({'url': result['foreign_landing_url'],
                        'title': result['title'],
                        'img_src': result['url'],
                        'template': 'images.html'})

    return results
```

Fichier settings.yml

Pour être pris en compte à l’initialisation d’une instance de Searx, un nouvel adaptateur doit être déclaré dans le fichier ***“searx/settings.yml”***.

Exemple pour CCengine :

```
  - name : ccengine
    engine : ccengine
    categories : images
    shortcut : cce
```

a)CC-engine

https://github.com/searx/searx/pull/2533

Ajout du moteur de recherche https://search.creativecommons.org/ comme nouvelle source d'images publiées sous licence Creative Commons.

b)Library of Congress

https://github.com/searx/searx/pull/2544

Ajout de la possibilité d'interroger le site de la bibliothèque du congrès américain pour des images.

c)Freesound

https://github.com/searx/searx/pull/2596

Un travail sonore recourt très fréquemment à la recherche de sons dans différentes banques de sons partagées sur internet. Pour autant, à l'heure actuelle il n'existe pas de moteur de recherche axé sur l'audio qui ne soit pas nécessairement de la musique.

Une des motivations était donc d'alimenter Searx avec des moteurs de recherche sonores des différentes plateformes qui permettront à terme de paramétrer Searx pour cet usage.

L'avancement de notre groupe sur l'ajout des *engines* a permis d'imaginer d'autres variantes de recherche, tels que l'intégration d'un lecteur audio dans les résultats permettant une pré-écoute des résultats. Enfin, l'API de Freesound étant très élaborée, ce fut l'occasion de se pencher sur la sémantique d'une recherche audio et sur la taxonomie des objets sonores. Par exemple, l'affichage des éléments jugés les plus pertinents dans les résultats: le choix de réduction de la description à 128 signes ou l'abandon de l'affichage du poids des fichiers qui ajoutait du "bruit" même après sa conversion d'octets en kilos, etc.

d)Artic (musée de Chicago)

https://github.com/searx/searx/pull/2665

Ajout de la source de données https://www.artic.edu/ , dans la catégorie « général » (ne traite pas les contenus images, ce qui fera éventuellement l'objet d'une autre PR) qui regroupe la totalité de la collection d'œuvres artistiques de l'« Art Institute of Chicago ». L'API est documentée de manière très complète sur http://api.artic.edu/docs/ .

e)Springer Nature

https://github.com/searx/searx/pull/2663

Springer est un important éditeur scientifique: https://www.springernature.com/, il met à disposition une API pour faire des recherches dans ses publications: https://dev.springernature.com/restfuloperations

L'utilisation de cette API requiert une clé, qu'on peut obtenir assez simplement via ce lien: https://dev.springernature.com/signup

Dans la *pull request*, on a commenté la clé API qui avait été utilisée lors du développement: en effet, si on a bien saisi, il revient à l'administrateur d'une instance Searx de se procurer les clés API et de les implémenter correctement.

f)Core.ac.uk

https://github.com/searx/searx/pull/2688

Core.ac.uk est un moteur qui référence la plus grande collection de papiers / articles / parutions de recherche en accès libre. Il est hébergé par The Open University et dispose de fonctions avancées de Text Mining.

6.Diverses installations de Searx sur serveur

Nous avons réalisé **trois types d'installations** du service Searx selon deux types d'usage :

- facilité d'usage pour un **auto-hébergement** « chez soi »:
  - Debian 10 et Yunohost
- **usage industriel** pour gérer un grand nombre de requêtes publiques :
  - Docker + Kubernetes
  - serveur dédié sous Debian 10 avec Apache

6.1.Docker et Kubernetes pour une instance publique

Il est possible d'installer Searx avec un docker-compose. Un dépôt est disponible avec les briques nécessaires (Searx, Morty, Filtron) à son implémentation sous docker. Il est conseillé d'utiliser Caddy comme serveur web, car il facilite la gestion des certificats pour une connexion sécurisée et facile à mettre en œuvre. L'intégralité du fichier docker compose est fournie en annexe.

Il faut souligner quelques points importants:

- un fichier .env qui contient les variables nécessaires

```
SEARX_HOSTNAME=${SEARX_HOSTNAME:-localhost}
SEARX_TLS=${LETSENCRYPT_EMAIL:-internal}
```

- Le bon mappage des ports, des URLS et la clé d'authentification HMAC pour Morty pour vérifier l'intégrité et l'authenticité des pages web :
  - Filtron :

```
"127.0.0.1:4040:4040"
"127.0.0.1:4041:4041"
listen 0.0.0.0:4040 -api 0.0.0.0:4041 -target searx:8080
```

  - Searx

```
BIND_ADDRESS=0.0.0.0:8080
BASE_URL=https://${SEARX_HOSTNAME:-localhost}/
```

  - Morty

```
"127.0.0.1:3000:3000"
MORTY_URL=https://${SEARX_HOSTNAME:-localhost}/morty/
MORTY_KEY=${MORTY_KEY}
```

Nous avons pu intégrer ce docker dans un cluster Kubernetes (6 x Raspberry, ubuntu server, K3S + containerd) de 2 façons:

- utilisation d'un fork de Searx qui est disponible dans un repo HELM sur Artifact Hub.

  La configuration se fait par le biais d'un fichier values.yml que l'on paramètre en fonction de nos besoins. Il va également installer Caddy comme serveur Web, Filtron, Morty et Searx suivant le même schéma que le docker-compose

- traduction du docker-compose avec Kompose pour avoir des fichiers exploitables sur le cluster.

La documentation pour l'installation sur un cluster Kubernetes est également dans les annexes.

6.2. Installation sur un serveur Debian

Nous avons réalisé une installation complète du service Searx sur un serveur privé virtuel (VPS[7]) hébergé chez l'entreprise OVH, puis avons fait de même sur une machine virtuelle d'un serveur sous NixOS hébergé et administré par deux bénévoles de l'association à but non lucratif Aquilenet.

a) Pré-requis

- Système d'exploitation de la machine virtuelle (dans les deux cas): Debian 10
- Avoir un nom de domaine qui pointe vers la machine virtuelle :
  - searx.ox2.fr => premier essai, machine consumée chez OVH qui n'a pas pu être récupérée à temps
  - searx-adsillh.melisse.org (contrainte actuelle: avoir une adresse IP publique en ipv6. Configuration selon le fournisseur d'accès internet[8])
- Mettre à jour les sources :
  - apt-get update
  - apt-get dist-uprade
- Installer **git** et **sudo** (apt-get install)
- Nommer le serveur comme son FQDN (dans notre cas : `hostnamectl set-hostname searx-adsillh.melisse.org`). C'est là-dessus que Searx va s'appuyer pour renseigner les variables durant son installation.

b) Serveur httpd Apache

Tout d'abord, il est nécessaire d'installer :

- un **serveur httpd**[9] permettant de servir des requêtes respectant le protocole de communication client-serveur HTTP entre notre serveur Searx et tous les utilisateurs qui se connecteraient à cette instance particulière de Searx;
- les **modules nécessaires** dans le cas pratique de l'application web Searx: headers, proxy, et proxy_http (cf. annexes).

c) Installation de l'application Searx

1. **Clonage du dépôt officiel** où le code source de l'application Searx est maintenu[10]
2. Lancement du **script d'installation** de Searx en mode administrateur[11] (cf annexes) :

   ```
   cd searx && sudo -H ./utils/searx.sh install all
   ```

   1. nous avons découvert un **bogue** au cours de cette procédure (que nous devons encore faire remonter aux mainteneurs du dépôt officiel).
   2. Résolution du bogue: créer un fichier *searx.service* dans */usr/lib/systemd/system/* pour pouvoir poursuivre avec la démonisation via uWSGI (cf. paragraphe suivant).

d) Installation de l'application uWSGI

Nous installons l'application uWSGI, qui permet de servir des applications écrites en Python, – application Searx qui utilise le framework python Flask dans notre cas – , en conjonction avec des serveurs web – serveur Apache dans notre cas –, selon la documentation officielle du projet Searx[12] (cf annexes).

7 https://en.wikipedia.org/wiki/Virtual_private_server
8 https://assistance.sfr.fr/internet-tel-fixe/box-nb6/activer-protocole-ipv6-box-sfr.html
9 Nous avons fait le choix du plus populaire parmi les serveurs httpd libres, à savoir Apache : https://www.apache.org/dyn/closer.cgi
10 https://github.com/searx/searx
11 https://searx.github.io/searx/admin/installation.html#installation-scripts
12 https://searx.github.io/searx/admin/installation-uwsgi.html#alltogether

On choisit le mode Emperor[13] car il permet de lancer plusieurs services applicatifs à la fois sur un même serveur http, ce qui permettrait éventuellement à Searx de cohabiter avec d'autres services applicatifs sur le serveur que nous utilisons pour administrer cette instance.

La procédure de vérification du résultat des différentes étapes d'installation est décrite en annexe:

- utilisation de netstat pour vérifier les flux actifs;
- liaison du proxy d'Apache2 à uWSGI;
- création du fichier de configuration *searx.conf* permettant de servir l'application Searx avec les bonnes informations de port et de socket, etc (cf. annexes).

e)Installation du reverse proxy HTTP Filtron

Filtron est un "reverse proxy HTTP" (basé sur le langage Go) qui permet de filtrer les requêtes faites entre clients et serveurs selon différentes règles qu'il prédéfinit. Dans notre cas, il permet d'honorer techniquement la promesse de respect des données à caractère personnel des utilisateurs finaux.

Il est utilisé entre le serveur web – ici Apache – et le serveur d'application – ici uWSGI –, lui-même connecté à l'application Searx sous Flask (cf annexes).

> **REMARQUE**: il est possible de s'épargner l'installation manuelle des 4 étapes précédentes, en exécutant directement le script filtron.sh : *sudo -H ./utils/filtron.sh install all* . Toutefois, avec le goût de comprendre précisément les outils techniques et l'infrastructure mobilisés par Searx, la description détaillée de ces étapes s'avère utile à la compréhension du projet.

f)Redirection de tous les flux en HTTPS

Cette étape consiste à obtenir des certificats SSL/TLS gratuits: dans notre cas, via le projet Certbot[14] développé par l'Electronic Frontier Foundation qui s'appuie sur l'autorité de certification à but non lucratif Letsencrypt[15] et à modifier le fichier de configuration d'Apache afin de rediriger de façon inconditionnelle les flux entrant sur le port 80 vers le port 443.

- Installation de Certbot (déployer des certificats Letsencrypt facilement. Développé par l'EFF : certbot.eff.org) : *sudo apt install python3-certbot-apache*

```
sudo apt install certbot
```

- Installation du plugin Certbot pour Apache :

```
sudo apt install python3-certbot-apache
```

- Arrêter le service Apache qui empêche Certbot d'écouter sur le port 80 le temps de la génération de certificats ;
- Faire générer les certificats par Certbot.

Le détail de l'enchaînement des commandes est disponible en annexes.

g)Le cas problématique de Morty

Nous avons installé Morty, un "result proxy" agissant comme service complémentaire de Searx, développé par le fondateur du projet[16], permettant de "nettoyer" les tags HTML malicieux retournés dans les résultats de recherche d'images et ainsi d'empêcher la divulgation d'information à des tiers.

Morty semble fonctionner, sans que nous comprenions comment l'implémenter dans l'architecture d'une instance publique. En effet, nous avons buté sur l'énigme suivante : 2 clients, à installation équivalente, pour le même terme recherché, n'étaient pas redirigés vers la même page de résultat.

Il est toutefois accessible en ajoutant « /morty » à la fin de l'url.

Il faudrait entamer des discussions avec les mainteneurs, pour en apprendre un peu plus sur le fonctionnement de Morty, puis éventuellement on pourrait proposer une PR sur le dépôt officiel du projet.

13 https://uwsgi-docs.readthedocs.io/en/latest/Emperor.html
14 https://certbot.eff.org/
15 https://letsencrypt.org/fr/
16 https://github.com/asciimoo/morty

6.3. Installation sur un serveur auto-hébergé avec Debian et Yunohost

Yunohost est basée sur Debian, elle permet de faire de l'auto-hébergement, à savoir d'héberger des services web sur une machine située chez soi. Searx est empaqueté pour Yunohost (https://yunohost.org/):

```
https://github.com/YunoHost-Apps/searx_ynh
Searx
Un méta-moteur de recherche respectueux de la vie privée et bidouillable
```

On a procédé à l'installation d'une instance de Searx.

Une première installation sur un de nos serveurs auto-hébergés a échoué. Une seconde tentative, faite quelques semaines plus tard sur le même serveur, a réussi (https://geolllibre.org/searx/): voilà qui fut une illustration de la sorte de «magie» des logiciels Libres dynamiques, qui «se» corrigent rapidement.

On est parti d'une instance de Yunohost personnelle, https://geolllibre.org/ , et on a choisi de faire l'installation en passant par l'interface web. Les détails de l'installation sont fournis en annexe.

À l'arrivée, une icône libellée « Se » permet, depuis la page d'accueil de Yunohost, de lancer Searx.

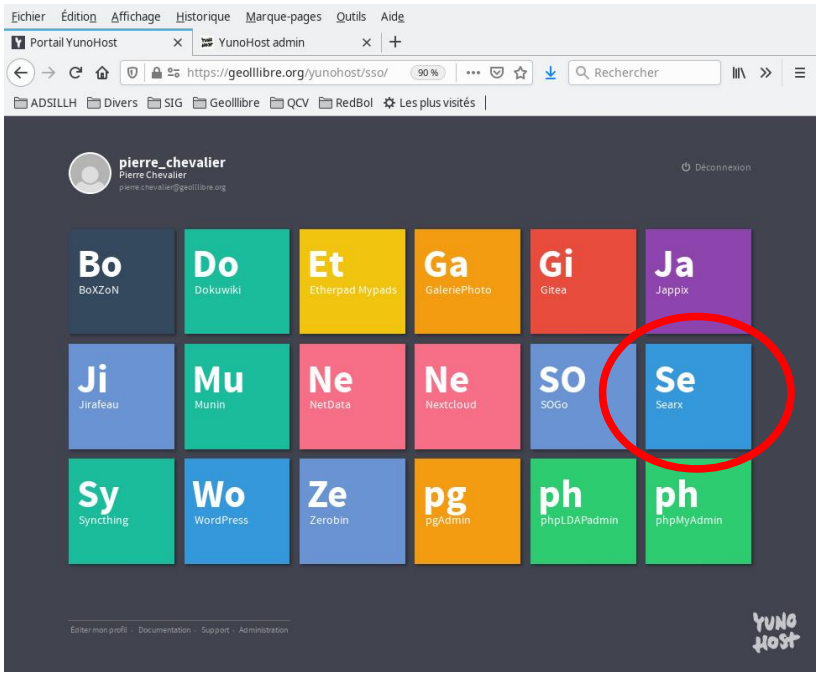


Fig. 7: Instance de Yunohost, avec Searx installé dessus

7. Prochaines étapes

- Poursuite du lien avec les bénévoles de l'association Aquilenet, afin d'administrer une à deux instances Searx (intégration de la documentation d'installation sous serveur dédié Debian dans leur wiki) ;
- Contributions :
  - implémentation d'une catégorie « sons » distincte de la catégorie « musique » et proposition d'une réflexion sur les modalités de recherche en prenant en compte des spécificités de métadonnées de sons.
  - en cours de validation : proposition d'ajout de 2 *engines* par Marc et Habib ;
  - en cours de travail-création :
    - 2 *pull requests* concernant la documentation:
      - « development quickstart » à compléter
      - correction de l'installation automatisée, amélioration du script `searx.sh`
    - 1 *pull request* concernant l'ajout de source de données (cf. sous-dossier /engines du dépôt searx) :
      - création d'un artic_images.py : pour les requêtes de fichiers images sur artic.edu

8.Conclusion

En résumé, il apparaît que malgré un bagage technique inégal au sein de notre groupe :

- on peut d'abord reconnaître que l'accueil des mainteneurs au sein du projet Searx (réactivité des réponses, aide à l'intégration même face à nos comportements de novices) a créé pour nous un cadre favorable au vu de l'expérience technique inégale des uns et des autres au sein de notre groupe ;
- une approche exploratoire façon « enquête criminelle » nous a tous permis de découvrir des liens entre divers objets du projet Searx (code, traduction ou documentation), et de proposer de les réparer ou de les améliorer ;
    - pour cela, un obstacle psychologique individuel récurrent à surmonter a étéindépendan celui d'accepter (ou non) de ne jamais avoir une vue complète du paysage technique du projet à un instant *t*, ainsi que la sensation concomitante d'y perdre parfois pied, notamment au début du projet tuteuré ;
    - lors des phases propédeutiques apparaissant au fil de l'eau (face à un nouveau langage, de nouveaux outils…), la présence d'un pair prenant temporairement un rôle de « tuteur » a souvent été fructueuse pour celui qui se trouvait face à une marche à gravir ;
- il paraît intéressant de relier la notion de « contribution » à celles de « persévérance » et de « pair-à-pair » à la fois :
    - en effet, nous avons pu observer qu'une forme d'équilibre dans le travail à deux ou trois individus à la fois (sans oublier tous les contributeurs inconnus qui fournissent des réponses dans les forums et autres blogs spécialisés) s'est souvent avéré d'une grande efficience dans les phases exploratoires du travail sur une contribution. D'un autre côté, la finalisation du travail d'une contribution se réalise seul sur sa machine (tests sur le fonctionnement du code, et make test) avec une relation directe avec les mainteneurs du projet ;
    - autrement dit, de ce que nous avons pu observer dans le cadre de ce projet tuteuré, dans le cycle de vie d'une contribution (du souhait de contribuer à la remise du résultat technique aux mainteneurs d'un projet donné), une « contribution individuelle » au sens de l'écosystème des logiciels libres semble souvent être une « contribution travaillée collectivement, et validée puis proposée individuellement ».

En conclusion, certains d'entre nous semblent avoir découvert un espace ludique suffisamment vaste et accueillant dans le cadre du projet Searx pour avoir envie d'y proposer de nouvelles contributions dans le futur.

Annexes

Annexe A - Structure de Searx

Structure générale

a)Webapp Searx, avec framework Flask

- Front end (UI) : /searx/templates (.html) , /searx/static (.css, .js, images)
- Back end (requêtes vers les APIs des moteurs de recherche, …) : /engines/[fichiers].py

b)Fichiers pour organiser les contributions au projet

- documentation développeurs /docs , /tests

c)Fichiers pour installer des instances Searx :

- documentation admin /docs et /dockerfiles

d)Personnalisation

- 2 thèmes[17] au choix pour les utilisateurs (/static/themes/oscar, /static/themes/simple…)
- des plugins (/plugins/infinite_scroll.py, …)

Il faut bien respecter la procédure et créer le bon environnement de travail développeur pour pouvoir participer au projet (voir la procédure sur https://searx.github.io). La procédure est assez simple, elle consiste en un git clone, un «sourçage» d'un fichier de configuration qui crée un environnement virtuel (virtualenv) pour Python.

```
mock==2.0.0
nose2[coverage_plugin]==0.9.2
cov-core==1.15.0
pycodestyle==2.6.0
pylint==2.4.4
plone.testing==5.0.0
splinter==0.11.0
transifex-client==0.12.2
unittest2==1.1.0
zope.testrunner==4.5.1
selenium==3.141.0
twine==3.2.0; python_version >= "3.6"
twine==1.15.0; python_version < "3.6"
Pallets-Sphinx-Themes==1.2.3
Sphinx==3.2.1; python_version >= '3.6'
Sphinx==3.0.1; python_version < '3.6'
sphinx-issues==1.2.0
sphinx-jinja==1.1.1
sphinx-tabs==1.3.0; python_version >= '3.6'
sphinx-tabs==1.1.13; python_version < '3.6'
sphinxcontrib-programoutput==0.16
sphinx-autobuild==2020.9.1; python_version >= '3.6'
sphinx-autobuild==0.7.1; python_version < '3.6'
linuxdoc @
git+http://github.com/return42/linuxdoc.git@70673dcf69e705e08d81f53794895dc15c4920b3#egg=linuxdoc
```

Fig. 8: Liste des dépendances nécessaires pour pouvoir contribuer au programme

17 Il y avait 5 thèmes, au début de notre contribution au projet.

Annexe B: nos participations

Procédure générale de contribution à Searx

Avant de contribuer à Searx, il faut paramétrer son environnement de développement. Voici les étapes que nous avons suivies :

1. Installer les pré-requis sur sa machine : Python3 & npm
2. Faire un Fork :
   - Depuis la page du dépôt de Searx, initier un Fork du projet, ce qui a pour effet de dupliquer les fichiers sur son dépôt personnel.
3. Clone de son fork localement :
   - Git clone mondépôt
4. Créer une branche :
   - Git checkout -b mabranche : c'est sur cette branche nouvellement créée que nous appliquons nos modifications.
5. Paramétrage de l'environnement de développement :
   - make pyenv : création du virtualenv
   - make install : installation des dépendances
6. Faire les tests : les tests vérifient le code la bonne syntaxe en s'appuyant sur le standard Pep8 pour Python[18]
   - make test
7. Initier la pull request :
   - après avoir mis à jour son dépôt avec un « git push », nous pouvons initier la Pull Request depuis l'interface de Github.

Documentation: Pull Requests 2273, 2276, 2277

Lors de la toute première prise de contact avec le logiciel Searx, en suivant les instructions d'installation de Searx, sur cette page https://searx.github.io/searx/admin/installation-searx.html#installation-basic , des erreurs, assez mineures, ont été détectées dans la documentation.

Voilà qui faisait une occasion parfaite de faire une toute première participation qui promettait d'être fort aisée. La suite des opérations infirma assez vite cette dernière assertion.

Le rapport de mi-parcours du Projet Tuteuré mentionnait les détails des opérations; on ne va pas les répéter ici, on va simplement les résumer.

On a fait le travail d'installation en suivant à la lettre les instructions de cette page: https://searx.github.io/searx/admin/installation-searx.html#installation-basic

Ce faisant, on a relevé trois incohérences, il s'agissait de quelques corrections à faire: un mkdir qui manquait, et une faute de frappe à xgd-open au lieu de xdg-open. Il y a aussi une commande sed qui était sans effet, mais c'est un détail non bloquant.

On a noté au passage que le script manage.sh n'a pas un comportement cohérent avec la documentation. Il est plus tard apparu qu'il était préférable d'avoir recours à des make plutôt qu'à ce script. Cela fait une possibilité de participation en plus, à ranger dans les idées à faire: rendre homogène la documentation avec l'utilisation (ou pas) du script manage.sh.

On a localisé les fichiers sources de la documentation, dans le même dépôt, dont on a fait un fork, un clone, on a créé une branche, puis on a édité les fichiers incriminés.

18 https://pep8.org/

Déroulement de la Pull Request, processus de validation par les mainteneurs du projet

Une première Pull Request (https://github.com/searx/searx/pull/2273) a été envoyée, mais bien trop vite, avant de rebaser sur le master upstream:

On a tenté d'amender la Pull Request déjà envoyée: cela s'est avéré bien plus complexe qu'il n'y paraissait; une seconde Pull Request a donc été préparée avec plus de soin, en repartant d'un clone tout frais, dans l'idée de supprimer la première PR.

Il a été amusant de constater, en révisant les modifications, qu'une des coquilles décelées avait entretemps été corrigée par quelqu'un d'autre dans une autre PR, la 2265 (https://github.com/searx/searx/pull/2265). Du point de vue de la gestion par git, cela n'a même pas généré de conflit.

Il y eut à ce moment un concours de circonstances assez improbable: la précédente PR 2273 n'avait fait l'objet d'aucune attention depuis deux jours, et je comptais la détruire, juste après avoir fait la PR 2276. Or voilà que, pendant les quelques secondes où ces deux PR coexistaient, return42 a révisé la PR 2273, se posant la question de la pertinence de la PR 2276.

En suivant ses conseils pour mieux rebaser, j'ai plutôt annulé la PR 2276. Finalement, c'est return42 qui a fait une autre Pull Request, la 2277, en y incorporant les modifications que j'avais suggérées.

À ce moment, j'ai voulu réviser cette Pull Request, en régénérant la documentation. En suivant les indications, j'ai repris la PR en question. Il a aussi fallu installer des dépendances, notamment du LaTeX.

La documentation générée en local ne comporte pas les boîtes à onglets où, précisément, figurent les textes corrigés. Après de nombreuses tentatives infructueuses de reconstruction de la documentation, j'ai validé «à l'aveuglette» la revue de la pull request 2277, et j'ai clôturé toutes mes pull requests devenues inutiles.

Il y a des dépendances qui ne figurent pas dans la documentation: il faudra, à terme, les y ajouter.

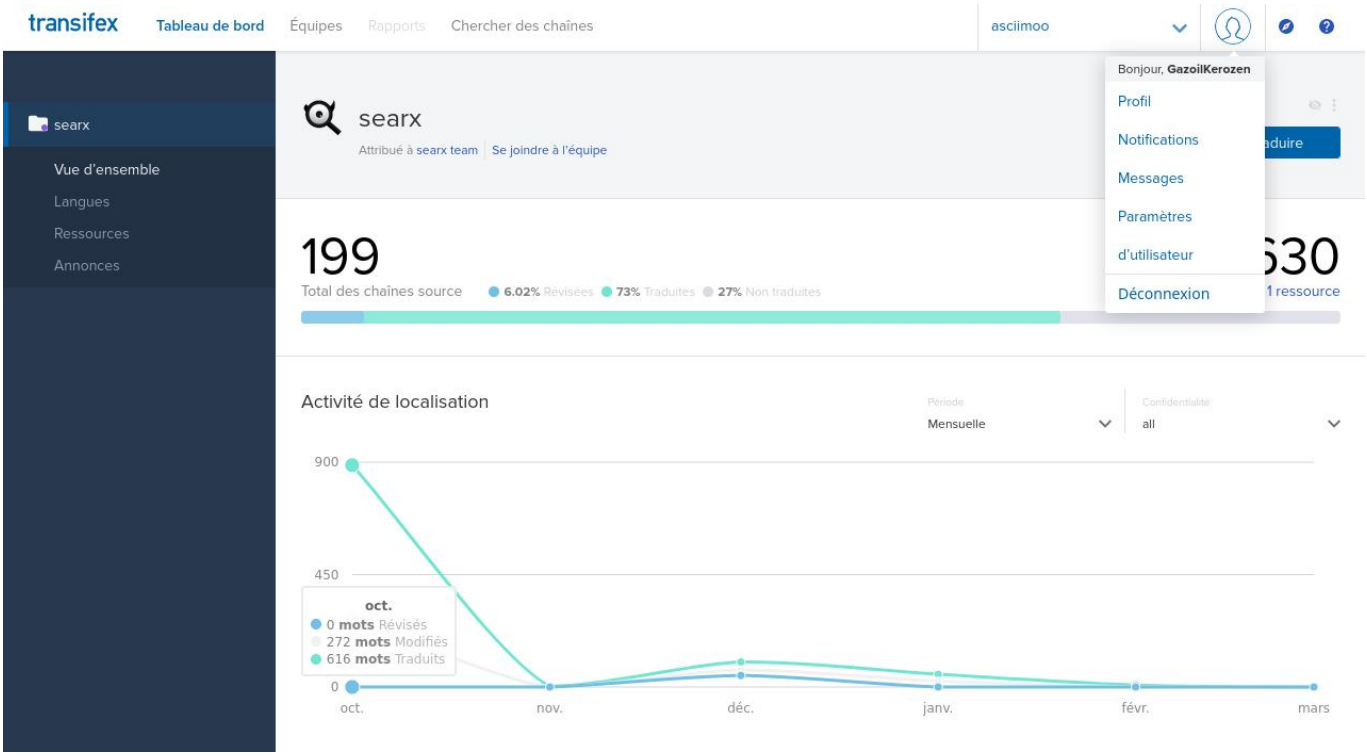
Documentation: Pull Request 2301

Fix typo virualenv #2301 https://github.com/searx/searx/pull/2301
En parcourant la documentation, le mot «virualenv» est apparu, et une rapide recherche a démontré qu'il ne s'agissait certainement pas d'une notion qui nous était jusqu'ici inconnue, mais plus certainement d'une faute d'orthographe au mot « virtualenv».

L'erreur se trouve dans des fichiers servant à la génération de la documentation. De toute évidence, la coquille a été faite une fois, puis a été recopiée ailleurs.

Il s'agissait d'une faute commise par return42, encore lui. Il y avait eu quelques changements récents, de sorte que la Pull Request avait besoin d'être rebasée. Ce fut chose aisée, avec l'aide de return42.

Traductions



Reset button: https://github.com/searx/searx/pull/2306

Solution apportée:

```
[gazoil@gazoil-cyborg searx]$ git show reset-butt
commit fc82e797b64c18f8cbcdb5280e4122c4961a7cf4 (HEAD -> reset-butt, origin/reset-butt)
Author: Gazoil <maildeguzel@gmail.com>
Date:   Mon Nov 30 15:24:16 2020 +0100

    hide the 'clear' button while no JS support

diff --git a/searx/templates/oscar/search.html b/searx/templates/oscar/search.html
index d853a8ec..2b3758ef 100644
--- a/searx/templates/oscar/search.html
+++ b/searx/templates/oscar/search.html
@@ -6,7 +6,7 @@
         <input type="search" autofocus name="q" class="form-control" id="q"
placeholder="{{ _('Search for...') }}" aria-label="{{ _('Search for...') }}" autocomplete="off"
value="{{ q }}" accesskey="s">
         <span class="input-group-btn">
             <button type="submit" class="btn btn-default" aria-label="{{ _('Start
search') }}"><span class="hide_if_nojs">{{ icon('search') }}</span><span class="hidden
active_if_nojs">{{ _('Start search') }}</span></button>
-            <button type="button" id="clear_search" class="btn btn-default" aria-
-label="{{ _('Clear search') }}"><span class="hide_if_nojs">{{ icon('remove') }}</span><span
class="hidden hide_if_nojs">{{ _('Clear') }}</span></button>
:...skipping...
commit fc82e797b64c18f8cbcdb5280e4122c4961a7cf4 (HEAD -> reset-butt, origin/reset-butt)
Author: Gazoil <maildeguzel@gmail.com>
Date:   Mon Nov 30 15:24:16 2020 +0100

    hide the 'clear' button while no JS support

diff --git a/searx/templates/oscar/search.html b/searx/templates/oscar/search.html
index d853a8ec..2b3758ef 100644
--- a/searx/templates/oscar/search.html
+++ b/searx/templates/oscar/search.html
@@ -6,7 +6,7 @@
         <input type="search" autofocus name="q" class="form-control" id="q"
placeholder="{{ _('Search for...') }}" aria-label="{{ _('Search for...') }}" autocomplete="off"
value="{{ q }}" accesskey="s">
         <span class="input-group-btn">
             <button type="submit" class="btn btn-default" aria-label="{{ _('Start
search') }}"><span class="hide_if_nojs">{{ icon('search') }}</span><span class="hidden
active_if_nojs">{{ _('Start search') }}</span></button>
-            <button type="button" id="clear_search" class="btn btn-default" aria-
```

```
label="{{ _('Clear search') }}"><span class="hide_if_nojs">{{ icon('remove') }}</span><span
class="hidden hide_if_nojs">{{ _('Clear') }}</span></button>
+           <button type="button" id="clear_search" class="btn btn-default hide_if_nojs" aria-
label="{{ _('Clear search') }}">{{ icon('remove') }}</button>
        </span>
      </div>
    </div>
~
```

Gruntfile: https://github.com/searx/searx/pull/2308

L'ajout d'une option dans gruntfile.js rajoute une ligne au fichier minifié.

Objectif

Modifier gruntfile.js du thème Oscar : sample grunt config

Voir la doc de Grunt Ici

Pour quoi faire ?

Une cartographie de code source (source map) est un fichier grâce auquel le débogueur peut faire le lien entre le code minifié étant exécuté et les fichiers sources originaux, permettant ainsi au débogueur (par exemple la console de développement de Firefox) de reconstruire la source originale et de l'afficher dans le débogueur. Voir MDN.

Fix invidious engine

Objectif

Faire en sorte que le moteur effectue les recherches sur des instances qui fonctionnent.

Voir Issue

Modifications

- J'arrive à faire un random sur la liste pour générer l'url de recherche
- *Problème* : l'url de recherche est défini au lancement de l'instance Searx.
- *Solution* : modification de `invidious.py` et `settings.yml` pour que `base_url` contienne plusieurs URL. Ainsi, le `random` est lancé à chaque recherche.

Cette première modification importante du code de Searx m'a permis de mieux appréhender le fonctionnement des moteurs de recherche de Searx.

Elle aura également été pour moi l'occasion de me rendre compte des possibilités de Git, et de me retrouver face à un mur après avoir fait des erreurs. N'arrivant pas à m'en sortir, j'ai finalement demandé aux mainteneurs de Searx de supprimer la Pull Request initiale pour en faire une propre (voir https://github.com/searx/searx/pull/2357).

Ajouts de moteurs de recherche

Freesound

Extrait du fichier engine:

```
"""
Freesound (Sound)
"""

from json import loads
from urllib.parse import urlencode

# about
```

```
about = {
        "website": &apos;https://freesound.org&apos;,
        "wikidata_id": &apos;Q835703&apos;
        "official_api_documentation": &apos;https://freesound.org/docs/api&apos;,
        "use_official_api": True,
        "require_api_key": False,
        "results": &apos;JSON&apos;,
}

# engine dependent config
categories = [&apos;music&apos;]
paging = True
```

```
# git show freesound
commit 321788f14a8a674984b9884094dd9626d6162a33 (HEAD -> master, freesound)
Merge: ffaf785f 90b9d0d6
Author: Alexandre Flament <alex@al-f.net>
Date:   Thu Feb 4 23:12:27 2021 +0100
Merge pull request #2528 from dalf/mod-ci-gh-pages
[mod] CI: minor changes
```

Springer Nature

API pour faire des recherches dans les publications de Springer Nature: https://dev.springernature.com/restfuloperations

Pour obtenir une clé pour utiliser l'API: https://dev.springernature.com/signup

En s'inspirant des pull requests des autres moteurs qui avaient été ajoutés à Searx, on a créé le fichier searx/engines/springer.py dont voici le contenu:

```
# pierre@latitude: ~/dev/searx < 2021_03_18__19:19:29 > [bashpid_4879 20] [springer_nature]
cat searx/engines/springer.py
# SPDX-License-Identifier: AGPL-3.0-or-later
"""
Springer Nature (science)
"""
from json import loads
from urllib.parse import urlencode

about = {
"website": 'https://www.springernature.com/',
"wikidata_id": 'Q21096327',
"official_api_documentation": 'https://dev.springernature.com/',
"use_official_api": True,
"require_api_key": True,
"results": 'JSON',
}

categories = ['science']
paging = True
nb_per_page = 10

# api_key = "a69685087d07eca9f13db62f65b8f601" #working API key, for test & debug
api_key = ""

base_url = 'https://api.springernature.com/metadata/json?'
search_string = "&{query}&s={page}&p={nb_per_page}&api_key={api_key}"


def request(query, params):
search_path = search_string.format(
query=urlencode({'q': query}),
nb_per_page=nb_per_page,
api_key=api_key,
page=params['pageno'])
params['url'] = base_url + search_path
return params


def response(resp):
results = []
json_data = loads(resp.text)
for record in json_data['records']:
results.append({
'title': record['title'],
'url': record['url'][0]['value']
})
```

```
return results
```

On a ajouté une entrée dans le fichier searx/settings.yml:

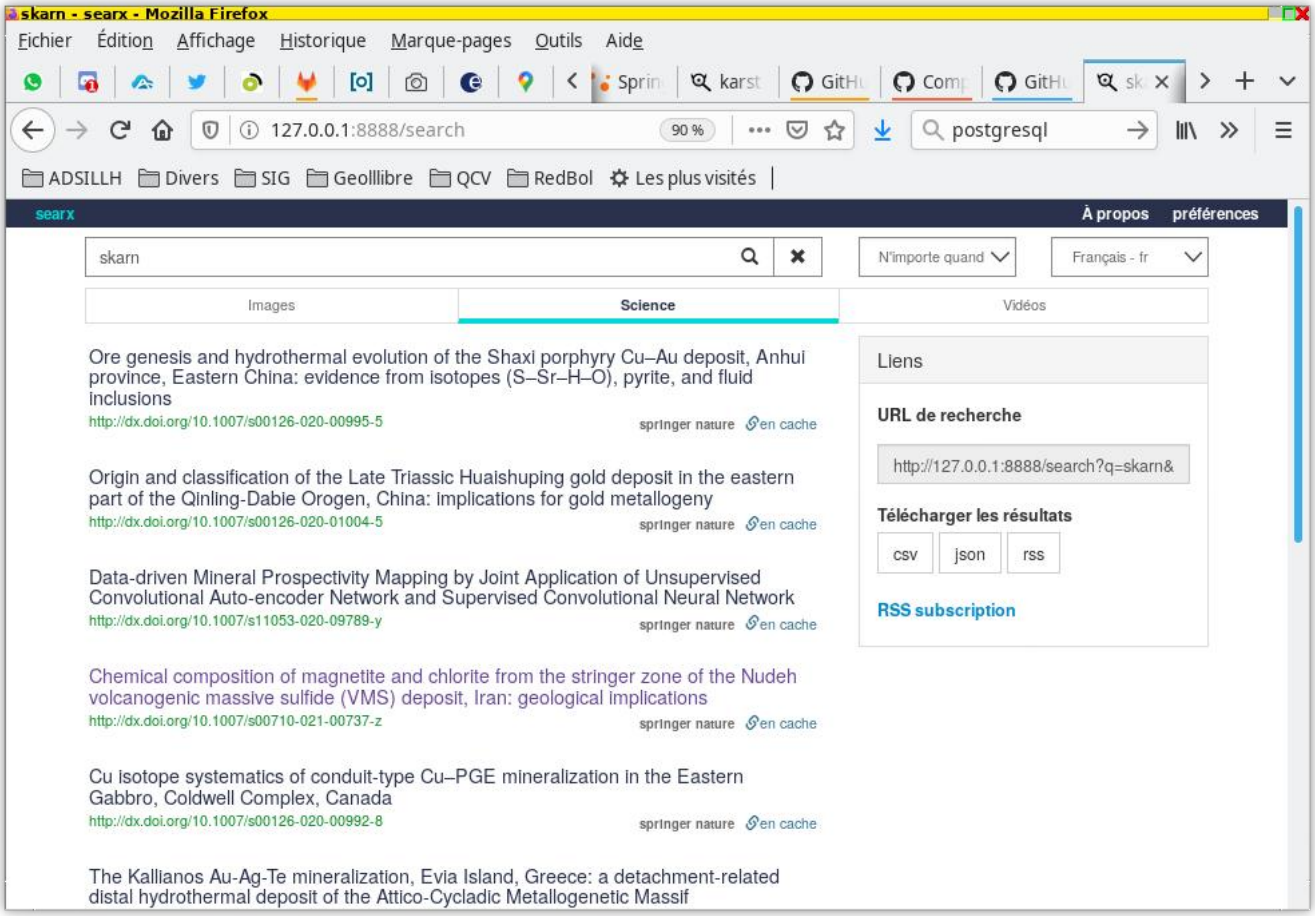
```
# pierre@latitude: ~/dev/searx < 2021_03_18__19:19:34 > [bashpid_4879 21] [springer_nature]
grep -i -A3 springer searx/settings.yml
- name : springer nature
engine : springer
shortcut : springer
categories : science
timeout : 6.0
```

Pour les tests, on a incorporé la clé API obtenue comme décrit plus haut. On a fait pas mal de tests avant d'aboutir au code python fonctionnel, notamment en interrogeant l'API par curl avec une redirection, puis en allant chercher dans le JSON obtenu. Voici les requêtes curl ayant donné satisfaction, sur lesquelles on a pu se baser:

```
# pierre@latitude: ~ < 2021_03_10__16:08:32 > [bashpid_20599 20]
curl "http://api.springernature.com/metadata/json?
q=skarn&api_key=a69685087d07eca9f13db62f65b8f601"
{"apiMessage":"This JSON was provided by Springer
Nature","query":"skarn","apiKey":"a69685087d07eca9f13db62f65b8f601","result":
[{"total":"2369","start":"1","pageLength":"10","recordsDisplayed":"10"}],"records":
[{"contentType":"Article","identifier":"doi:10.1007/s12517-021-06884-z","url":
[{"format":"","platform":"","value":"http://dx.doi.org/10.1007/s12517-021-06884-
z"}],"title":"Geochronology, petrogenesis and tectonic importance of Eocene I-type magmatism in
the Eastern Pontides, NE Turkey","creators":[{ "creator":"Vural, Alaaddin" },
{ "creator":"Kaygusuz, Abdullah" }],"publicationName":"Arabian Journal of
Geosciences","openaccess":"false","doi":"10.1007/s12517-021-06884-
z","publisher":"Springer","publicationDate":"2021-03-
10","publicationType":"Journal","issn":"1866-7538","volume":"14","number":"6","genre":
["OriginalPaper","Original
Paper"],"startingPage":"1","endingPage":"24","journalId":"12517","copyright":"©2021 Saudi Society
for Geosciences","abstract":"Eastern Pontides are hosted to many granitoids with different comp
[...]
# pierre@latitude: ~ < 2021_03_10__16:08:32 > [bashpid_20599 21]
curl "http://api.springernature.com/metadata/json?
q=skarn&api_key=a69685087d07eca9f13db62f65b8f601" | json_pp | head -20
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 32170 100 32170 0 0 74640 0 --:--:-- --:--:-- --:--:-- 74813
{
"records" : [
{
"volume" : "56",
"copyright" : "Â©2020 Springer-Verlag GmbH Germany, part of Springer Nature",
"creators" : [
{
"creator" : "Wang, Shiwei"
},
{
"creator" : "Zhou, Taofa"
},
{
"creator" : "Hollings, Pete"
},
{
"creator" : "Yuan, Feng"
},
{
"creator" : "Fan, Yu"
[...]
```

Au moment de faire la pull-request, on a commenté la ligne comportant la clé API, car, si l'on a bien compris, c'est à la personne qui maintient une instance search de se charger d'obtenir les clés des différentes API.

Les tests faits dans l'environnement de développement (avec make run) étaient satisfaisants, voici un résultat:



La pull-request soumise est la suivante: https://github.com/searx/searx/pull/2663
Les tests opérés sur github, ainsi que make test, ont révélé quelques anomalies d'indentations, des lignes manquantes avant les def, et une ligne vide en trop en fin de fichier: on a corrigé cela et re-poussé les modifications en faisant:

```
git push --set-upstream origin springer_nature -f
```

L'accueil de la contribution fut enthousiaste, de par la qualité de la base de Springer; return42 a suggéré de mettre en place une pagination des résultats, ce qui fut fait avec quelques commits supplémentaires. Ces modifications sont incluses dans le code mentionné plus haut.
Par la suite, il a travaillé dans son dépôt et il a ajouté des champs au résultat final. On en est resté au stade où il faut que je récupère ses deux commits et que je les intègre dans ma PR.

Annexe C: installations de Searx dans divers environnements

Environnement de développement

1. Forker le projet Searx sur son dépôt github personnel (en appuyant sur le bouton Fork sur l'interface github par exemple)
2. Faire une branche et ne travailler **que sur celle-ci**
3. Faire la pull-request depuis **sa** branche en n'ayant **qu'un seul commit**

```
$ make pyenv
PYENV     usage: source ./local/py3/bin/activate
...

$ make install
PYENV     usage: source ./local/py3/bin/activate
PYENV     using virtualenv from ./local/py3
PYENV     install .

$ make test
build searx/brand.py
build utils/brand.env
PYENV     usage: source ./local/py3/bin/activate
PYENV     using virtualenv from ./local/py3
LINT      test.pylint
...

$ make run
PYENV     usage: source ./local/py3/bin/activate
PYENV     install .
./local/py3/bin/python ./searx/webapp.py

//Pour compiler les themes
$ make themes
```

Docker et Kubernetes

On a procédé à un déploiement en utilisant Docker-compose avec filtron et morty, en utilisant caddy comme serveur HTTPS.

Les variables d'environnement sont définies dans un fichier à part.

```
version: '3.7'

services:

caddy:
container_name: caddy
 image: caddy:2-alpine
 network_mode: host
 volumes:
 - ./Caddyfile:/etc/caddy/Caddyfile:ro
 - caddy-data:/data:rw
 - caddy-config:/config:rw
 environment:
 - SEARX_HOSTNAME=${SEARX_HOSTNAME:-localhost}
 - SEARX_TLS=${LETSENCRYPT_EMAIL:-internal}
 cap_drop:
 - ALL
 cap_add:
 - NET_BIND_SERVICE
 - DAC_OVERRIDE

 filtron:
 container_name: filtron
 image: dalf/filtron
 restart: always
 ports:
 - "127.0.0.1:4040:4040"
 - "127.0.0.1:4041:4041"
 networks:
 - searx
 command: -listen 0.0.0.0:4040 -api 0.0.0.0:4041 -target searx:8080
 volumes:
 - ./rules.json:/etc/filtron/rules.json:rw
 read_only: true
```

```
cap_drop:
- ALL

searx:
container_name: searx
image: searx/searx:latest
restart: always
networks:
- searx
command: ${SEARX_COMMAND:-}
volumes:
- ./searx:/etc/searx:rw
environment:
- BIND_ADDRESS=0.0.0.0:8080
- BASE_URL=https://${SEARX_HOSTNAME:-localhost}/
- MORTY_URL=https://${SEARX_HOSTNAME:-localhost}/morty/
- MORTY_KEY=${MORTY_KEY}
cap_drop:
- ALL
cap_add:
- CHOWN
- SETGID
- SETUID
- DAC_OVERRIDE

morty:
container_name: morty
image: dalf/morty
restart: always
ports:
- "127.0.0.1:3000:3000"
networks:
- searx
command: -timeout 6 -ipv6
environment:
- MORTY_KEY=${MORTY_KEY}
- MORTY_ADDRESS=0.0.0.0:3000
logging:
driver: none
read_only: true
cap_drop:
- ALL

networks:
searx:
ipam:
driver: default

volumes:
caddy-data:
caddy-config:
```

Pour ce qui concerne nstallation Kubernetes (j'ai encore quelques problemes pour les certificats let's encrypt avec kubernetes), j'ai trouvé un Helm Chart pour Searx pour l'installer avec déjà presque tout de configuré correctement:

```
helm repo add k8s-at-home https://k8s-at-home.com/charts/
helm repo update
helm install searx k8s-at-home/searx -f values.yaml
```

Le fichier values.yaml permet de paramétrer Searx selon nos besoins.

J'ai rajouté:

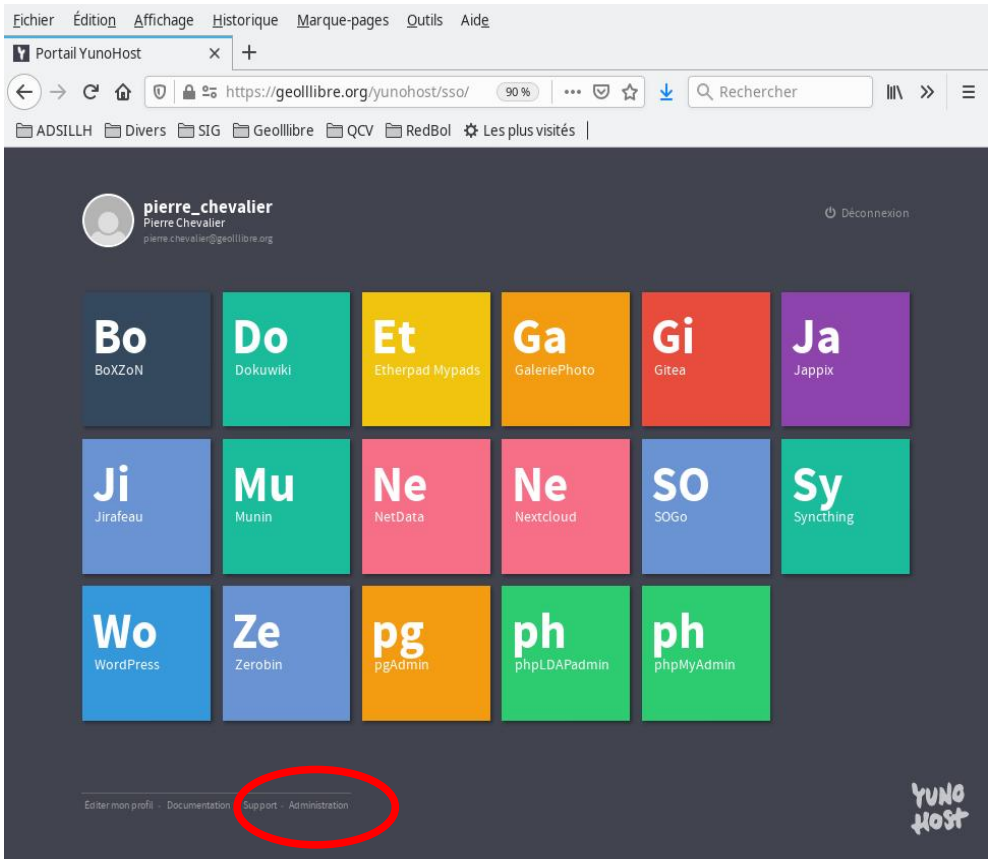
```
searx.baseUrl string "https://searx.marc-cenon.com"
```

Serveur auto-hébergé avec Debian et Yunohost

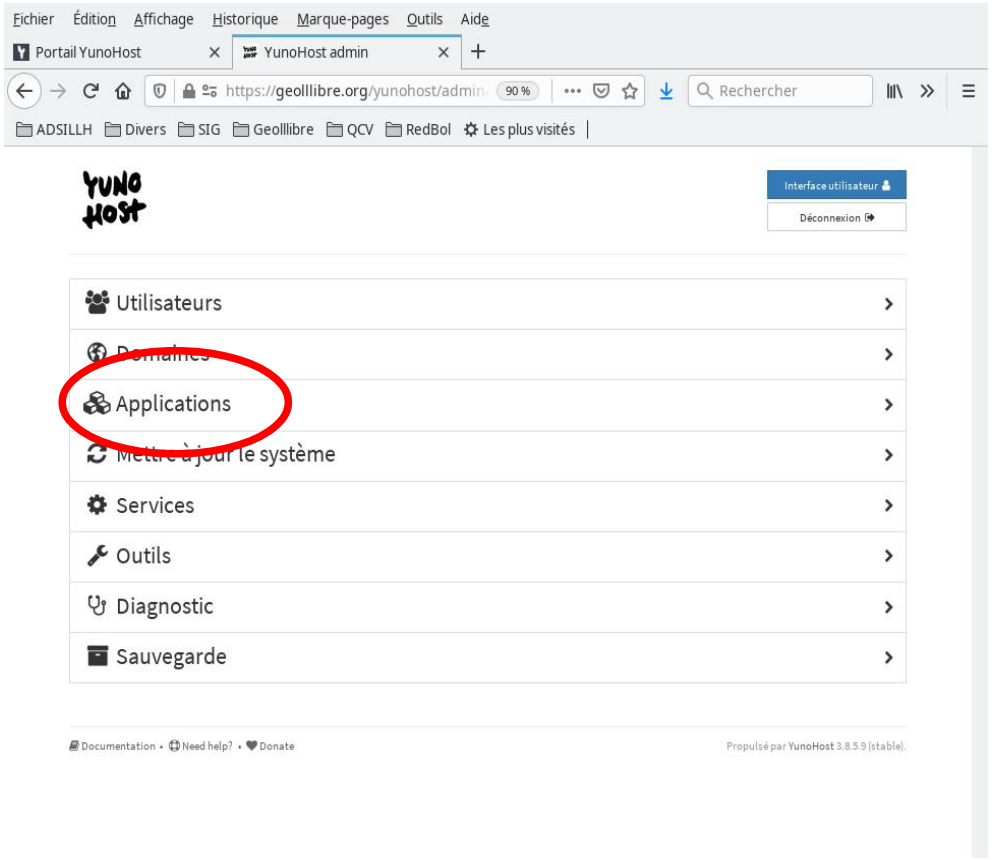
Yunohost est basée sur Debian, elle permet de faire de l'auto-hébergement, à savoir d'héberger des services web sur une machine située chez soi.

Parmi les applications qui sont disponibles dans yunohost figure Searx. On a procédé à l'installation d'une instance de Searx, comme on va le voir.

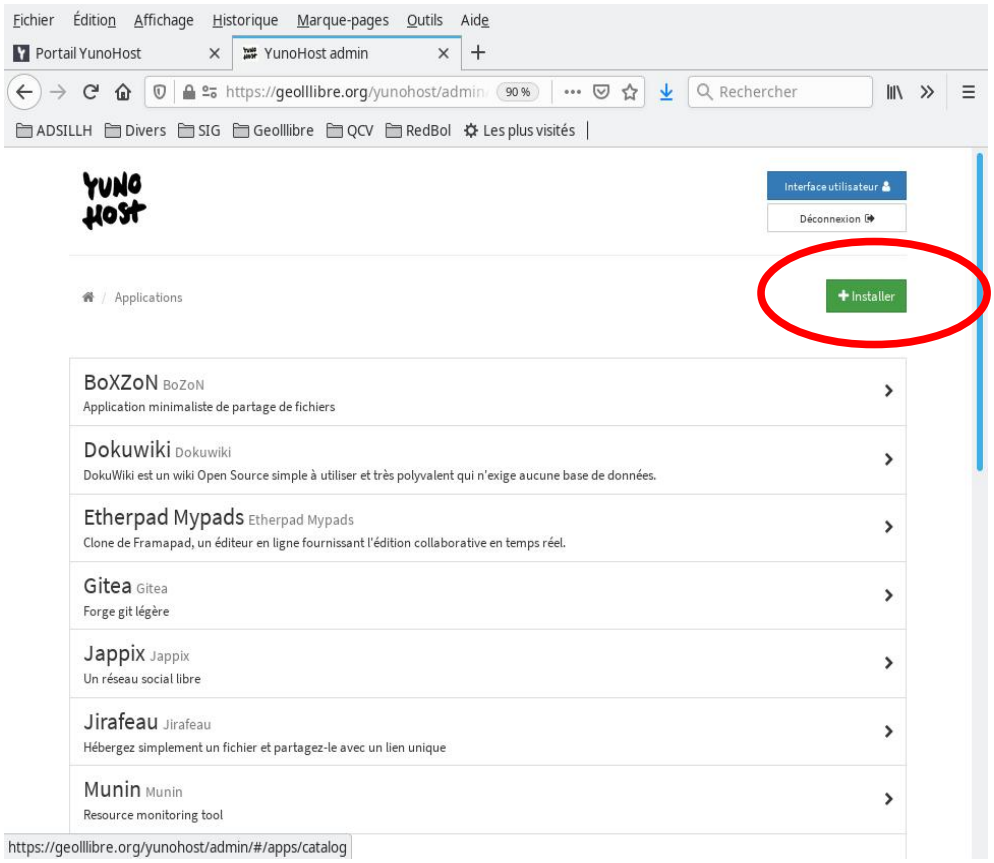
On part d'une instance de Yunohost personnelle, https://geolllibre.org/ , et on a choisi de faire l'installation en passant par l'interface web:



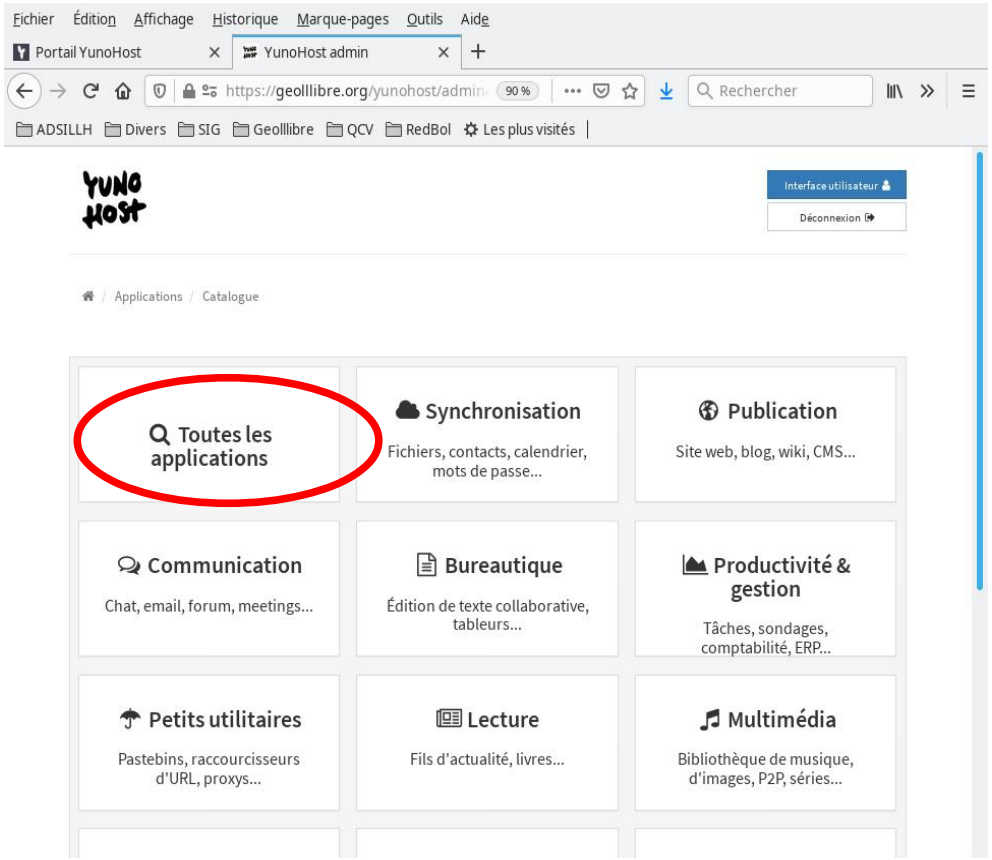
Il faut passer par l'interface d'administration, où il faut choisir les applications:



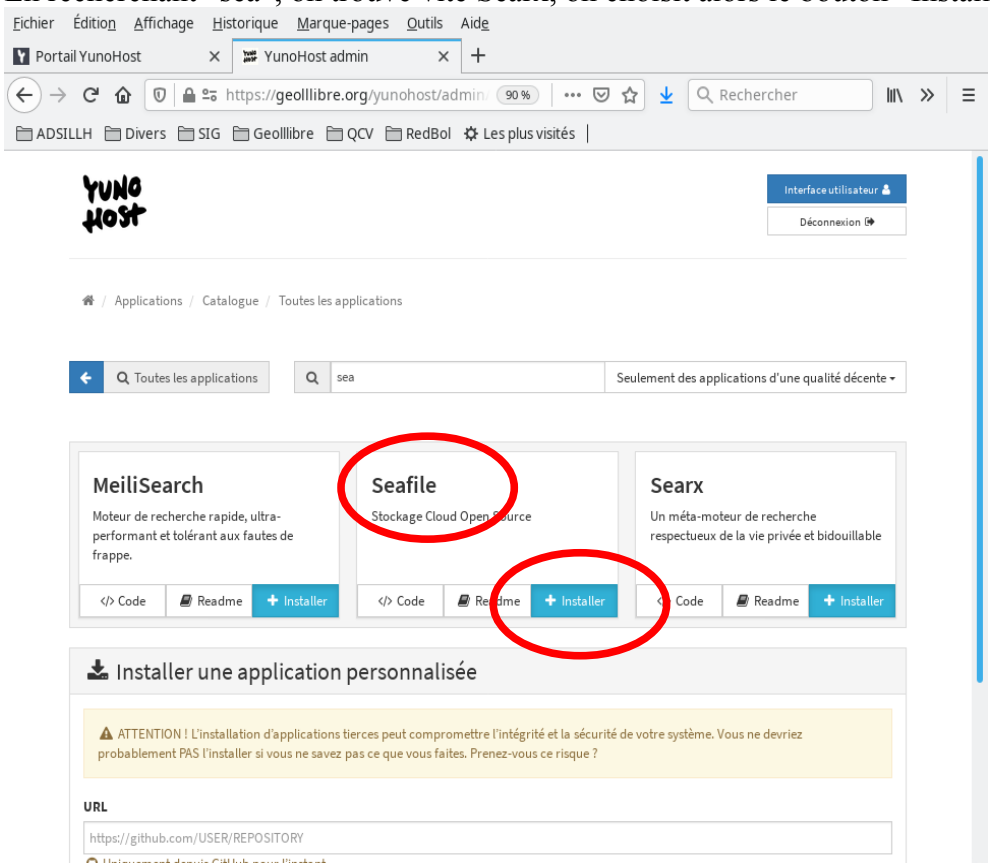
On choisit ensuite le bouton “Installer”:



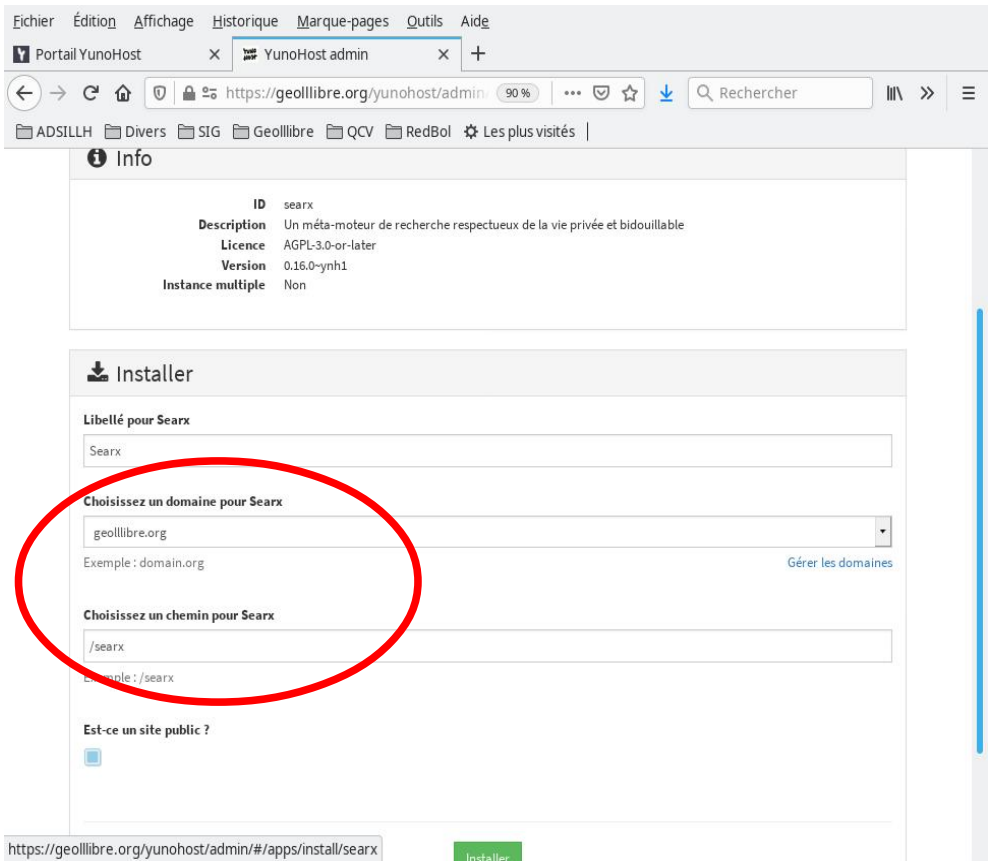
Il faut ensuite choisir le bouton recherchant “Toutes les applications”:



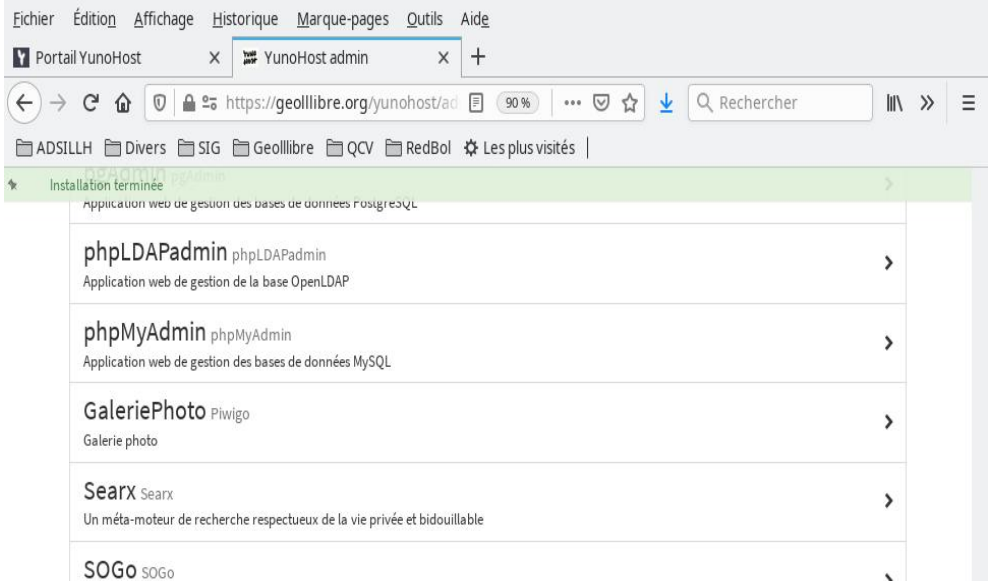
En recherchant “sea”, on trouve vite Searx, on choisit alors le bouton “Installer”:



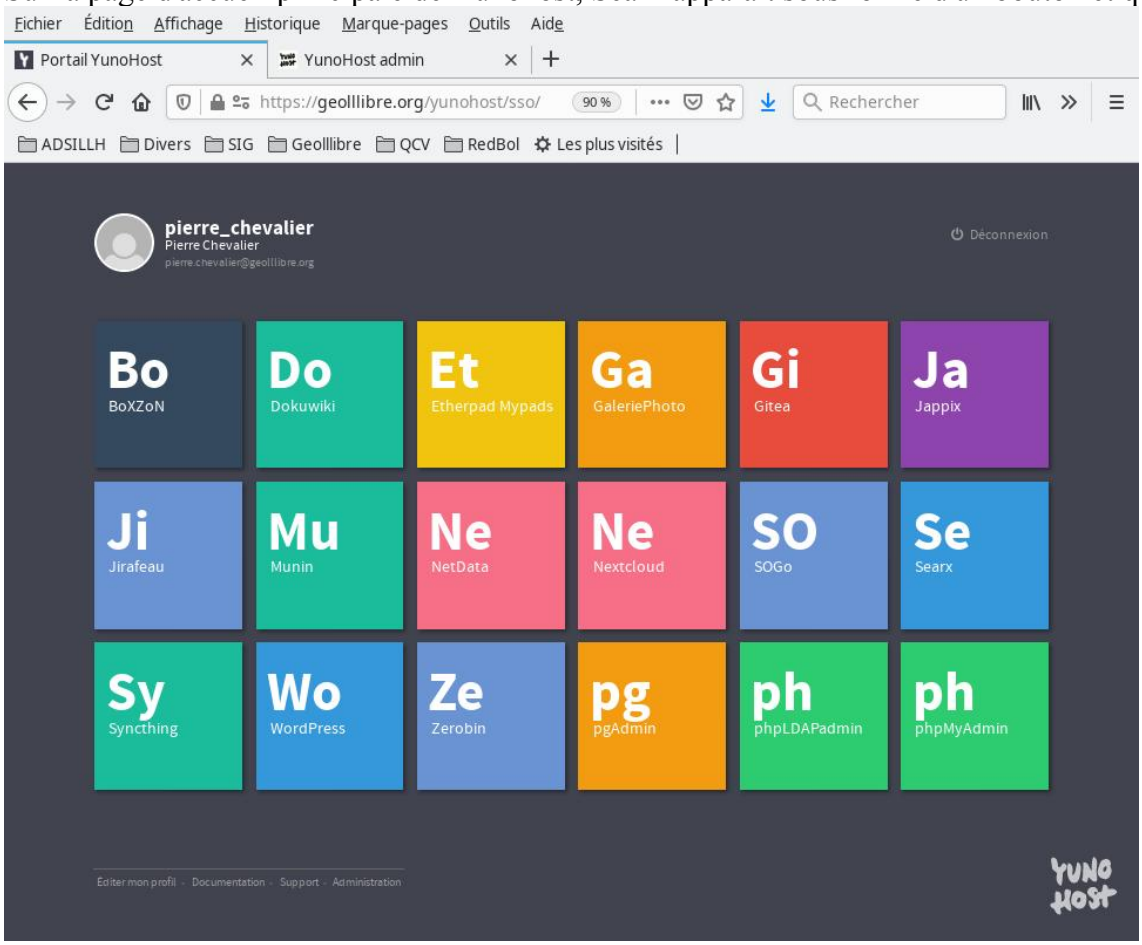
Dans les options d'installation, on choisit le domaine principal, et on peut au besoin spécifier un chemin différent:



Une fois l'installation terminée, Searx apparaît dans la liste des applications:

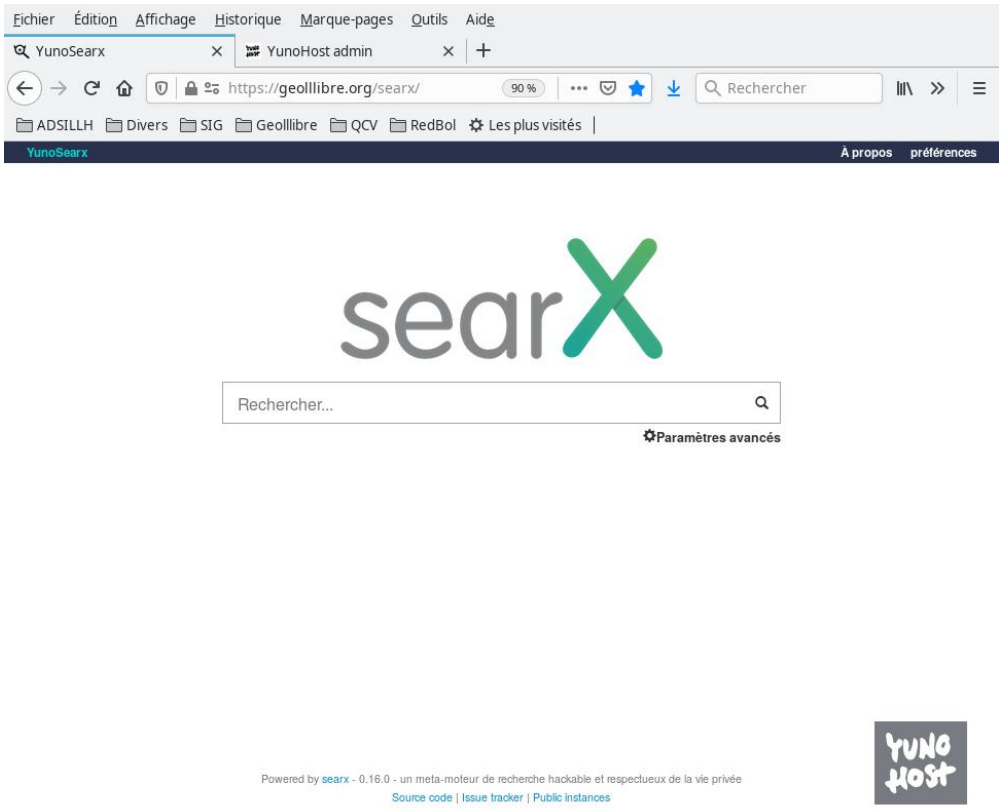


Sur la page d'accueil principale de Yunohost, Searx apparaît sous forme d'un bouton étiqueté “Se”:



Actionner sur ce bouton lance Searx, tout comme l’URL https://geolllibre.org/searx/:

L’aisance de l’installation de cette application sur Yunohost force l’admiration de ceux qui font tout le travail d’empaquetage des applications pour Yunohost.



Installation sur un serveur dédié Debian

a)Pré-requis

(optionnel) Détail des commandes pour éditer les fichiers de “host” pour lancer des commandes plus faciles à retenir :

1. `sudo hostnamectl set-hostname searx.ox2.fr`
2. Éditer le fichier hosts :

   ```
   vi /etc/hosts
   127.0.0.1 searx.ox2.fr searx
   ```

3. Vérifier le résultat :

   ```
   hostnamectl status
   ```

b)Serveur http Apache

Tout d’abord, il est nécessaire d’installer un serveur httpd (nous avons fait le choix du plus populaire parmi les serveurs httpd libres : Apache 2), et d’activer les modules nécessaires avec la commande « a2enmod »:

```
sudo -H apt-get install apache2
sudo -H a2enmod headers
sudo -H a2enmod proxy
sudo -H a2enmod proxy_http
```

c)Installation de l’application Searx

1. Cloner le dépôt de Searx dans le dossier de son choix :

   ```
   git clone https://github.com/searx/searx searx
   ```

2. `cd searx`
3. Lancer le script pour installer Searx (pré-requis : avoir le paquet ‘curl’ installé):

   ```
   sudo -H ./utils/searx.sh install all
   ```

NB: le script ci-dessus permet de réaliser en 1 ligne de commande l'ensemble des opérations suivantes : https://searx.github.io/searx/admin/installation-searx.html#installation-basic

4. `sudo -H ./utils/searx.sh activate service`

   Renvoie le message d'erreur :

   `Err: Unit searx.service could not be found.`

   (bogue expliqué dans le corps du rapport)

d)Installation de l'application uWSGI

Selon la source : https://searx.github.io/searx/admin/installation-uwsgi.html#alltogether

1. `cd /etc/uwsgi/apps-available`
2. `ls`

   On observe alors que le fichier `searx.ini` est bien créé

   On choisit le mode Emperor car il permet de lancer plusieurs services applicatifs à la fois sur un même serveur httpd (et pas un seul):

   https://uwsgi-docs.readthedocs.io/en/latest/Emperor.html

3. En faisant
   1. `netstat -l`

   on peut voir tous les services actifs à un instant t

   2. `sudo -H apt-get install libapache2-mod-proxy-uwsgi`
4. On crée le fichier `/etc/apache2/sites-available/searx.conf` :

```
LoadModule headers_module /usr/lib/apache2/mod_headers.so
LoadModule proxy_module /usr/lib/apache2/modules/mod_proxy.so
LoadModule proxy_uwsgi_module /usr/lib/apache2/modules/mod_proxy_uwsgi.so

ServerName searx-adsillh.melisse.org

# SetEnvIf Request_URI /searx dontlog
# CustomLog /dev/null combined env=dontlog

ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined

<Location />

Require all granted
Order deny,allow
Deny from all
# Allow from fd00::/8 192.168.0.0/16 fe80::/10 127.0.0.0/8 ::1
Allow from all

ProxyPreserveHost On
ProxyPass unix:/run/uwsgi/app/searx/socket|uwsgi://uwsgi-uds-searx/

</Location>
```

   On observe que le wsgi tourne sur un socket dédié : `/run/uwsgi/app/searx/socket`

```
$netstat -l
unix  2      [ ACC ]     STREAM     LISTENING     58013    /run/uwsgi/app/searx/socket
```

5. Activation searx.conf :

   `sudo a2ensite searx.conf`

6. Désactivation site par défaut :

   `sudo a2dissite 000-default.conf`

Si vous obtenez le message d'erreur suivant (sur searx-adsillh.melisse.org en l'occurrence) ,

`curl http://searx-adsillh.melisse.org`

```
curl: (7) Couldn’t connect to server
```

alors c’est qu’il vous est nécessaire de configurer une adresse ipv6 auprès de votre fournisseur d’accès internet.

e)Installation du reverse proxy HTTP Filtron

1. Créer le fichier `/etc/systemd/system/filtron.service`

```
[Unit]
Description=filtron
After=syslog.target
After=network.target

[Service]
Type=simple
User=filtron
Group=filtron
WorkingDirectory=/usr/local/filtron
ExecStart=/usr/local/filtron/go-apps/bin/filtron -api '127.0.0.1:4005' -listen '127.0.0.1:4004' -
rules '/etc/filtron/rules.json' -target '127.0.0.1:8888'

Restart=always

Environment=
USER=filtron
HOME=/usr/local/filtron # Some distributions may not support these hardening directives. If you
cannot # start the service due to an unknown option, comment out the ones not supported # by your
version of systemd.

ProtectSystem=full
PrivateDevices=yes
PrivateTmp=yes
NoNewPrivileges=true

[Install]
WantedBy=multi-user.target
```

2. Activer le service filtron :

```
sudo systemctl enable filtron.service
```

3. Daemoniser le service filtron pour qu’il soit actif en permanence :

```
sudo systemctl start filtron.service
```

4. Modifier le `/etc/apache2/sites-available/searx.conf` , puis le réactiver.

REMARQUE: Une autre possibilité est de lancer le script d’installation avec filtron dès l’étape a :

1. `cd <chemin-searx>/searx/`
2. `sudo -H ./utils/filtron.sh install all`

Cela fait toutes les installations nécessaires (apache, uwsgi) et crée directement le fichier `searx.conf` avec la configuration de filtron incluse :

```
# -*- coding: utf-8; mode: apache -*-

LoadModule headers_module       /usr/lib/apache2/modules/mod_headers.so
LoadModule proxy_module         /usr/lib/apache2/modules/mod_proxy.so
LoadModule proxy_http_module    /usr/lib/apache2/modules/mod_proxy_http.so
#LoadModule setenvif_module      /usr/lib/apache2/modules/mod_setenvif.so

# SetEnvIf Request_URI /searx dontlog
# CustomLog /dev/null combined env=dontlog

# SecRuleRemoveById 981054
# SecRuleRemoveById 981059
# SecRuleRemoveById 981060
# SecRuleRemoveById 950907

<Location / >

    <IfModule mod_security2.c>
        SecRuleEngine Off
    </IfModule>
```

```
    Require all granted

    Order deny,allow
    Deny from all
    #Allow from fd00::/8 192.168.0.0/16 fe80::/10 127.0.0.0/8 ::1
    Allow from all

    ProxyPreserveHost On
    ProxyPass http://127.0.0.1:4004
    RequestHeader set X-Script-Name /searx

</Location>
```

---

5. Pour servir Searx, supprimer ou renommer le fichier `/var/www/index.html` vers lequel le serveur Apache pointe par défaut :

   ```
   mv /var/www/index.html /var/www/index.html.bak
   ```

6. Ainsi, dans le fichier `searx.conf` , on peut remplacer `<Location /searx>` par `<Location />` pour avoir l'app servie sur `http://<servername>/` au lieu de `http://<servername>/searx` comme le propose la documentation officielle de Searx.
7. Redémarrer le serveur apache pour que toutes ces modifications soient prises en compte:

   ```
   sudo systemctl restart apache2.service
   ```

f)Redirection de tous les flux en HTTPS

- Installation de Certbot (déployer des certificats Letsencrypt facilement. Développé par l'EFF: https://certbot.eff.org) et du plugin Certbot pour Apache:

  ```
  sudo apt install certbot python3-certbot-apache
  ```

- Arrêter le service Apache qui empêche Certbot d'écouter sur le port 80 :
- Faire générer les certificats par Certbot

  (Répondre à la question : accepter ou non de partager son adresse email avec l' EFF pour recevoir leur newsletter)

  ```
  sudo systemctl stop apache2.service
  ```

  ```
  sudo certbot certonly --standalone
  ```

- Les certificats sont générés dans:

  `/etc/letsencrypt/live/searx-adsillh.melisse.org :`

```
$ tree /etc/letsencrypt/live/searx-adsillh.melisse.org /etc/letsencrypt/live/searx-
adsillh.melisse.org
├── cert.pem -> ../../archive/searx-adsillh.melisse.org/cert1.pem
├── chain.pem -> ../../archive/searx-adsillh.melisse.org/chain1.pem
├── fullchain.pem -> ../../archive/searx-adsillh.melisse.org/fullchain1.pem
├── privkey.pem -> ../../archive/searx-adsillh.melisse.org/privkey1.pem
└── README
```

- Relancer le service Apache :
  - `sudo systemctl start apache2.service`
- Modification du fichier de conf :

  `/etc/apache2/sites-available/searx.conf`

```
<VirtualHost *:80>

ServerName searx-adsillh.melisse.org
Redirect permanent / https://searx-adsillh.melisse.org

</VirtualHost>

<VirtualHost *:443>
```

```
LoadModule headers_module       /usr/lib/apache2/modules/mod_headers.so
LoadModule proxy_module         /usr/lib/apache2/modules/mod_proxy.so
LoadModule proxy_http_module    /usr/lib/apache2/modules/mod_proxy_http.so
#LoadModule setenvif_module      /usr/lib/apache2/modules/mod_setenvif.so

ServerName searx-adsillh.melisse.org

SSLProxyEngine on

SSLCertificateFile /etc/letsencrypt/live/searx-adsillh.melisse.org/fullchain.pem
SSLCertificateKeyFile /etc/letsencrypt/live/searx-adsillh.melisse.org/privkey.pem
Include /etc/letsencrypt/options-ssl-apache.conf

SSLProtocol all -SSLv2 -SSLv3 -TLSv1 -TLSv1.1
SSLHonorCipherOrder on
SSLCipherSuite "EECDH+ECDSA+AESGCM EECDH+aRSA+AESGCM EECDH+ECDSA+SHA384 EECDH+ECDSA+SHA256
EECDH+aRSA+SHA384 EECDH+aRSA+SHA256 EECDH+aRSA+RC4 EECDH EDH+aRSA RC4 !aNULL !eNULL !LOW !3DES !
MD5 !EXP !PSK !SRP !DSS !RC4"
Header always set Strict-Transport-Security "max-age=63072000; includeSubDomains"
# SetEnvIf Request_URI /searx dontlog
# CustomLog /dev/null combined env=dontlog

# SecRuleRemoveById 981054
# SecRuleRemoveById 981059
# SecRuleRemoveById 981060
# SecRuleRemoveById 950907

  <Proxy *>
    Require all granted
  </Proxy>

<Location />

    <IfModule mod_security2.c>
        SecRuleEngine Off
    </IfModule>

    Require all granted

    Order deny,allow
    Deny from all
    #Allow from fd00::/8 192.168.0.0/16 fe80::/10 127.0.0.0/8 ::1
    Allow from all

    ProxyPreserveHost On
    ProxyPass http://localhost:4004/
    ProxyPassReverse http://localhost:4004/
    #RequestHeader set X-Script-Name /searx
    RequestHeader set X-Forwarded-Proto "https"

</Location>
</VirtualHost>
```

- Activer les modules rewrite et ssl dans apache : `sudo a2enmode` rewrite et `sudo a2enmode ssl`.
- `systemctl status apache2` pour vérifier si le service apache tourne bien.
- On redémarra le service Apache : `sudo systemctl restart apache2`
- On regarde si tous les flux passent bien en https sur la console dans le navigateur à l'adresse searx-adsillh.melisse.org :

Et voilà :