Logiciels

- Créer une boutique en ligne avec Wordpress
- Gitlab
- Gramps Web
- RequestTracker
- TinyTinyRSS : rétablir la consultation web des articles publiés (+ bonus)
- Une web radio avec MPD
- Vaultwarden
- Wisemapping

Créer une boutique en ligne avec Wordpress

On commence par installer un Wordpress. Je ne vais pas détailler, c’est un sujet fort bien traité sur les Internetz.

Les extensions

Pour transformer Wordpress en boutique en ligne, il faut installer quelques modules. Mais avant ça, voici ceux que j'ai installé pour la sécurité, les performances... :

- Antispam Bee, pour l'antispam ;
- iThemes Security, pour la sécurité ;
- Smush, pour réduire la taille des images ;
- WP Super Cache, pour les performances.

Ensuite, des extensions pour améliorer la boutique, mais pas obligatoires :

- Contact Form 7, pour le formulaire de contact ;
- Google XML Sitemaps, pour améliorer le référencement (normalement plus nécessaire depuis quelques versions de Wordpress) ;
- Mastodon Autopost, pour pouetter automatiquement quand on ajoute un article à la boutique.

Voici les extensions pour la boutique elle-même :

- WooCommerce, l'extension principale, proposée par la boîte qui développe Wordpress, donc ça va, j'ai assez confiance sur la compatibilité ;
- WooCommerce Blocks, pour faire fonctionner WooCommerce avec le nouvel éditeur de Wordpress ;
- WooSwipe, pour avoir une galerie d'image sur les pages des produits (quand on clique sur l'image pour zoomer, pis pour voir les autres images, vous voyez le genre) ;
- WooCommerce Weight Based Shipping, pour gérer les frais d'envoi selon le poids de la commande ;
- Payment Gateway Based Fees and Discounts for WooCommerce, pour pouvoir ajouter des frais selon la méthode de paiement choisie.

Le thème

Il va de soit qu'un thème pour un blog a peu de chance d'aller pour faire une boutique. J'ai choisi le thème Shop Isle, que je trouve simple et bien foutu. J'en ai néanmoins fait un thème enfant pour dégager ces cochonneries de google fonts.

Les frais d'envoi

L'extension « WooCommerce Weight Based Shipping » me permet de définir des frais d'envoi selon le poids de la commande ainsi que sa destination (vous imaginez bien qu'envoyer deux cartes postales en France ne coûte pas le même prix qu'en envoyer 15 dans un autre pays).

Cette extension est parfaite, à l'exception de l'interface pour choisir les pays de destination quand on crée les règles : ça va bien quand on ajoute une ou deux destinations, mais pas quand on doit en sélectionner 12 (france métropolitaine + DOM/TOM) ou plus (les pays de l'UE).

Personnellement, j'ai choisi de faire des frais d'envoi à prix plus ou moins coûtant : le tarif des timbres, arrondi un peu au-dessus pour faire le prix de l'enveloppe. Je ne dis pas que je ne fais pas un peu de bénéfice dessus, mais ça se compte à coup de centimes.

Le paiement

WooCommerce embarque déjà plusieurs moyens de paiements, je dois dire que c'est très bien foutu !

J'ai activé le virement SEPA, parce que ça ne coûte rien, ni à l'envoyeur, ni au destinataire. Par contre, comme ça prend du temps (le virement prend généralement 24h pour apparaître sur les comptes, sans compter que certaines banques mettent plusieurs jours pour accepter un nouvel IBAN et permettre des virements vers celui-ci), j'ai ajouté la possibilité d'utiliser Paypal : ça propose d'utiliser son compte Paypal (et c'est assez répandu pour que ça soit pratique pour un paquet de gens) pour payer ou la carte bancaire. Par contre, ça m'a fait deux blagounettes :

- pour effectivement pouvoir récupérer les paiements, j'ai du passer mon compte en compte *business*. Rien de bien méchant, mais la récupération des sous ne fonctionnait pas et rien n'indiquait sur Paypal que c'était ce qu'il fallait faire (il y avait juste un message d'erreur complètement inutile). Une fois mon compte passé en *business*, plus de souci ;
- mais du coup, des frais s'appliquent (pas pour le client, qui paie bien ce que ma boutique lui indiquait, mais pour moi : je ne récupère pas autant d'argent que ce qu'a payé le

client). Comme je ne souhaite pas en être de ma poche, j'ai installé « Payment Gateway Based Fees and Discounts for WooCommerce » qui me permet d'ajouter le pourcentage (3,4%) et le forfait (0,25€) de la commission lorsqu'on choisit Paypal pour payer.

Conditions générales de vente

Pour les rédiger, j'ai d'abord cherché sur le web, et je suis tombé sur https://www.donneespersonnelles.fr/cgv, qui propose carrément une extension Wordpress. Après l'avoir installée, je me suis contenté de repomper les CGV qu'elle proposait : ça m'a permis, tout d'abord, de les lire, et ensuite de corriger des typos.

Après quoi, j'ai désinstallé l'extension □□

Conclusion

Je ne pense pas qu'il soit nécessaire pour moi de trop détailler le réglage des différentes extensions : les infos se trouvent assez facilement sur le web, et même si les réglages sont touffus, c'est assez intuitif. La plus grosse partie du temps fut perdue dans la recherche des extensions et du thème kivonbien.

Après quelques tâtonnements, la mise en place de ma boutique fut assez simple. J'espère que cet article servira à d'autres pour leur éviter de chercher autant que moi.

Gitlab

Recalculer la taille d’un dépôt

Tapez ceci dans la console rails (`gitlab-rails console`) :

```
project = Project.find_by_id(24495)
pp project.statistics.repository_size
pp project.repository.size
pp project.repository._uncached_size
project.repository.expire_all_method_caches
pp project.repository.size
project.statistics.refresh!
```

Recalculer la taille des artefacts

Il y a parfois un bug avec le calcul de la taille des projets, comme par exemple des artefacts virés (pour cause de suppression des *pipelines* afférents) dont le poids n’a pas été enlevé de la taille du projet.

Pour recalculer la taille des artefacts d’un projets, tapez ceci dans la console rails (`gitlab-rails console`) :

```
project = Project.find_by_id(23002)
stat = project.statistics.build_artifacts_size
old = project.builds.sum(:artifacts_size).to_i
real = Ci::JobArtifact.artifacts_size_for(project).to_i
diff = real + old - stat
puts "#{Time.now} : ID #{project.id} => stat = #{stat}; old = #{old}; real = #{real}; diff =
#{diff}"
ProjectStatistics.increment_statistic(project, :build_artifacts_size, diff)
```

Méthode trouvée sur https://gitlab.com/gitlab-org/gitlab-foss/-/merge_requests/20697#note_91526778

Gramps Web

Gramps Web est un logiciel de généalogie basé sur Gramps et interopérable avec celui-ci.

Son installation est normalement basée sur Docker mais comme j'ai horreur de cette technologie, j'ai décidé de l'installer à la main.

Gramps et Gramps Web utilisent normalement des bases SQLite mais on peut leur faire utiliser des bases PostgreSQL. Cependant, lorsque j'ai tenté ceci, Gramps n'a pas utilisé une base dédiée mais a créé des tables… je ne sais pas très bien où. Bref, cela ne m'a pas l'air très propre (et ce n'est pas grave si on utilise Docker, j'en conviens) et je ne voulais pas pourrir mon serveur PostgreSQL, donc SQLite ira très bien.

Attention, petit bug avec les gestionnaires de mot de passe

Si vous n'arrivez pas à vous connecter alors que votre gestionnaire de mot de passe remplit bien les champs, c'est à cause de ce bug. Il suffit de modifier l'entrée à la main (un caractère dans chaque champ, et on l'efface) et ça fonctionne.

Dépendances

```
apt install gramps graphviz gir1.2-gexiv2-0.10 gir1.2-osmgpsmap-1.0 python3-icu python3-virtualenv
```

Sur votre machine

Installez `gramps`, lancez et créez un arbre familial. Comme nom, j'ai utilisé mon nom de famille, pour ce tutoriel, considérons qu'il s'agit de `Foobar`.

Le lancement de gramps et la création de l'arbre familial ont créé un dossier dans `~/.gramps/`. Copiez ce dossier sur votre serveur web :

```
scp -r ~/.gramps votre_serveur:
```

Création de dossiers divers

```
mv ~votre_user_de_connexion/.gramps/ ~www-data/
mkdir -p /var/www/gramps/config \
          /var/www/gramps/static \
          /var/www/gramps/db \
          /var/www/gramps/media \
          /var/www/gramps/indexdir \
          /var/www/gramps/users \
          /var/www/gramps/thumbnail_cache \
          /var/www/gramps/cache \
          /var/www/gramps/cache/reports \
          /var/www/gramps/cache/export \
          /var/www/gramps/tmp \
          /var/www/gramps/persist \
          /var/www/gramps/secret
touch /var/www/gramps/static/index.html
chown -R www-data: /var/www/gramps/ /var/www/.gramps/
```

Installation d’un virtualenv

```
sudo -u www-data -s /bin/bash
```

Puis, en tant que `www-data`

```
cd ~/gramps
virtualenv venv
. ./venv/bin/activate
pip install gunicorn gramps-webapi
if [ ! -s /var/www/gramps/secret/secret ]; then
    python3 -c "import secrets;print(secrets.token_urlsafe(32))"  | tr -d "\n" >
/var/www/gramps/secret/secret
fi
```

```
GRAMPSWEB_SECRET_KEY=$(cat /var/www/gramps/secret/secret)
cat <<EOF >config/config.cfg
TREE="Foobar"
DISABLE_AUTH=False
SECRET_KEY="$GRAMPSWEB_SECRET_KEY"
MEDIA_BASE_DIR="/var/www/gramps/media"
SEARCH_INDEX_DIR="/var/www/gramps/indexdir"
STATIC_PATH="/var/www/gramps/static"
BASE_URL="https://gramps.example.org"
EMAIL_HOST="127.0.0.1"
EMAIL_PORT="25"
EMAIL_USE_TLS=False
DEFAULT_FROM_EMAIL="no-reply+gramps@example.org"
THUMBNAIL_CACHE_CONFIG__CACHE_DIR="/var/www/gramps/thumbnail_cache"
USER_DB_URI="sqlite:////var/www/gramps/users/users.sqlite"
REPORT_DIR="/var/www/gramps/cache/reports"
EXPORT_DIR="/var/www/gramps/cache/export"
EOF

export PYTHONPATH="/var/www/gramps/venv/lib/python3.11/site-packages:/usr/lib/python3/dist-
packages"
if [ -z "$(ls -A /var/www/gramps/indexdir)" ]; then
    python3 -m gramps_webapi --config /var/www/gramps/config/config.cfg search index-full
fi

wget https://github.com/gramps-project/gramps-webapi/archive/refs/tags/v1.4.0.tar.gz \
      https://github.com/gramps-project/Gramps.js/releases/download/v23.11.0/grampsjs-
v23.11.0.tar.gz

tar xvf v1.4.0.tar.gz
ln -s gramps-webapi-1.4.0/ gramps-webapi
cd gramps-webapi
python3 -m gramps_webapi --config /var/www/gramps/config/config.cfg user migrate

tar xvf grampsjs-v23.11.0.tar.gz
ln -s grampsjs-v23.11.0 grampsjs

rm v1.4.0.tar.gz grampsjs-v23.11.0.tar.gz
exit
```

Nginx

Voici la configuration Nginx que j'utilise, tirée de https://github.com/gramps-project/Gramps.js/blob/main/default.conf.template, à mettre dans `/etc/nginx/sites-available/gramps.example.org`.

```
server {
    listen 80;
    listen [::]:80;
    listen 443 http2 ssl;
    listen [::]:443 http2 ssl;

    server_name gramps.example.org;

    ssl_certificate     /etc/letsencrypt/live/gramps.example.org/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/gramps.example.org/privkey.pem;

    access_log  /var/log/nginx/gramps.example.org.access.log;
    error_log   /var/log/nginx/gramps.example.org.error.log;

    index index.html;
    root /var/www/gramps/grampsjs;

    location / {
        try_files $uri $uri/ $uri.html /index.html;
    }

    location /api {
        add_header       "Access-Control-Allow-Origin" $http_origin;
        proxy_redirect   off;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;

        # WebSocket
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
```

```
        proxy_set_header Connection "upgrade";

        proxy_pass http://127.0.0.1:5000;
    }
}
```

Puis :

```
ln -s ../sites-available/gramps.example.org /etc/nginx/sites-enabled
nginx -t && nginx -s reload
```

Systemd

Voici le service systemd que j'utilise, à mettre dans `/etc/systemd/system/gramps-webapi.service` :

```
[Unit]
Description=Shortened URLs service
Requires=network.target postgresql.service
After=network.target postgresql.service

[Service]
User=www-data
EnvironmentFile=/etc/default/gramps-webapi
WorkingDirectory=/var/www/gramps
ExecStart=/var/www/gramps/venv/bin/gunicorn -w $GUNICORN_NUM_WORKERS -b 127.0.0.1:5000
gramps_webapi.wsgi:app --timeout 120 --limit-request-line 8190
SyslogIdentifier=gramps-webapi
Restart=always

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

Et dans `/etc/default/gramps-webapi` :

```
GRAMPS_API_CONFIG=/var/www/gramps/config/config.cfg
PYTHONPATH="/var/www/gramps/venv/lib/python3.11/site-packages:/usr/lib/python3/dist-packages"
GUNICORN_NUM_WORKERS=2
```

Créez les fichiers puis :

```
systemctl daemon-reload
systemctl enable --now gramps-webapi
```

Première connexion

Connectez-vous à votre serveur, créez l’utilisateur administrateur et normalement, c’est tout bon !

RequestTracker

RequestTracker (RT) est un outil de tickets extrêmement puissant et flexible. Tellement flexible qu'on en vient à se prendre la tête pour faire des trucs de oufs.

Petit tour des trucs que j'ai mis en place sur un RT 4.4.0.

Utiliser le *plus addressing*

De façon simple, pour que RT traite les tickets qui arrivent sur l'adresse email dédiée, on la met dans le `/etc/aliases` de sa machine. Ça fait un truc comme ça :

```
rt:         "|/opt/rt4/bin/rt-mailgate --queue general --action correspond --url
https://rt.example.org/"
rt-comment: "|/opt/rt4/bin/rt-mailgate --queue general --action comment --url
https://rt.example.org/"
```

Vous noterez que cela met les mails à destination de cette adresse dans la *queue* (ou file) `general`. Or on utilise généralement plus d'une *queue* dans un système de ticket : cela permet de diriger automatiquement vers les personnes les plus à même d'y répondre.

Le problème avec ce système, c'est qu'il faudrait ajouter une adresse dédiée à chaque fois que l'on crée une nouvelle file. Ça peut vite devenir usant.

On va donc utiliser le *plus addressing*. Cette technique, toute bête, consiste à ajouter un discriminant à une adresse mail, précédé généralement d'un `+` (mais on peut configurer son serveur de mail pour utiliser n'importe quel caractère). `rt@example.org` aura alors pour alias (par exemple) `rt+file_bidule@example.org`.

Pour que RT puisse utiliser cette technique, il faut ajouter `--extension=queue` dans la commande du `/etc/aliases` :

```
rt:         "|/opt/rt4/bin/rt-mailgate --extension=queue --queue general --action correspond
--url https://rt.example.org/"
rt-comment: "|/opt/rt4/bin/rt-mailgate --extension=queue --queue general --action comment --
url https://rt.example.org/"
```

Voilà ! Il ne vous reste plus qu'à créer vos files via l'interface web. Attention, créez-les avec un nom qui passera dans une adresse mail. Pas d'espace par exemple.

RT récupérera le nom de la file kivabien dans la partie entre le `+` et le `@` et placera automatiquement le ticket dans cette file, tout en gardant la file `general` par défaut.

Les articles

Quoi de plus casse-pieds que de se répéter encore et encore en donnant toujours la même réponse ? Heureusement il y a les articles qui peuvent vous servir de réponses pré-enregistrées :-)

Création des classes et des articles

Allez dans le menu `Administration > Articles > Classes > Ajouter`, créez vos classes (j'en ai créé une par file, n'ayant pas réussi à assigner automatiquement des articles aux files), cochez `Tous les articles de cette classe doivent être disponibles sous forme de liste sur la page de réponse d'un ticket`, décochez `Inclure le résumé de l'article` et `Inclure le nom de l'article` et cochez `Include le champs personnalisé 'Content' > Value` (oh la belle typo de traduction) qui apparaîtra après avoir enregistré la classe (pour ces trois derniers, vous faites comme vous le sentez hein).

Liez la classe à une file via le menu `S'applique à`.

Voilà, vous n'avez plus qu'à créer vos articles dans la classe que vous venez de créer via le menu `Articles > Ajouter`.

Et là, magie, lorsque vous répondez via l'interface web, vous pourrez choisir une réponse pré-enregistrée.

Placement des articles dans la réponse

Je suis un grand fan du *bottom-posting*, mais RT place l'article au-dessus de la citation du message précédent. Remédions à cela.

```
cd /opt/rt4
mkdir -p local/html/Elements/
cp share/html/Elements/MessageBox local/html/Elements/
vi local/html/Elements/MessageBox
```

Cherchez la ligne contenant

```
% $m->comp('/Articles/Elements/IncludeArticle', %ARGS) if $IncludeArticle;
```

et remplacez-la par

```
% if ($IncludeArticle) {
%    my $article = $m->scomp('/Articles/Elements/IncludeArticle', %ARGS);
%    $article    =~ s{\n}{<br />}g;
%    $article    = RT::Interface::Email::ConvertHTMLToText($article);
%    $Default   .= $article unless ($Default =~ s/(.*)(-- .*)/$1$article$2/m);
% }
```

Hop ! votre article se trouve maintenant entre la citation et votre signature :-)

(un redémarrage de RT est peut-être nécessaire pour que cela soit pris en compte)

Ajout des articles pertinents dans le mail de notification d'un nouveau message

Une des forces de RT est de permettre aux intervenants de répondre aux tickets par mail. Le problème est que cela empêche de piocher dans les réponses pré-enregistrées.

Qu'à cela ne tienne, ajoutons-les au mail de notification envoyé aux membres du support.

Allez dans `Administration > Global > Modèles > Choisir`. Il faut modifier le modèle `Notification de modification HTML` (oui, j'ai traduit le nom de mes modèles, mais il est simple à repérer, il est utilisé par les *scrips* 8 et 11).

Ajoutez ceci en bas du modèle :

```
{ my $hotlist = RT::Articles->new( RT->SystemUser );
  $hotlist->LimitHotlistClasses;
  $hotlist->LimitAppliedClasses( Queue => $Ticket->QueueObj );
  my $content   = "-- \n<p><b>Réponses pré-enregistrées pour cette catégorie de
tickets:</b></p>";

  if ($hotlist->Count) {
    while (my $article = $hotlist->Next) {
      $content .= '<p><b>'.$article->Name.'</b><br/>';
      my $class   = $article->ClassObj;
      my $cfs     = $class->ArticleCustomFields;
      my %include = (Name => 1, Summary => 1);
      $include{"CF-Title-".$_->Id} = $include{"CF-Value-".$_->Id} = 1 while $_ = $cfs->Next;
      $include{$_} = not $class->FirstAttribute("Skip-$_") for keys %include;
```

```
      while (my $cf = $cfs->Next) {
        next unless $include{"CF-Title-".$cf->Id} or $include{"CF-Value-".$cf->Id};
        my $values = $article->CustomFieldValues($cf->Id);
        if ($values->Count == 1) {
          my $value = $values->First;
          if ($value && $include{"CF-Value-".$cf->Id}) {
            $content .= '<br/>';
            my $c     = $value->Content || $value->LargeContent;
            $c =~ s/\r?\n/<br\/>/g;
            $content .= $c;
          }
        } else {
          my $val = $values->Next;
          if ($val && $include{"CF-Value-".$cf->Id}) {
            $content .= '<br/>';
            my $c     = $value->Content || $value->LargeContent;
            $c =~ s/\r?\n/<br\/>/g;
            $content .= $c;
          }
          while ($val = $values->Next) {
            if ($include{"CF-Value-".$cf->Id}) {
              $content .= '<br/>';
              my $c     = $value->Content || $value->LargeContent;
              $c =~ s/\r?\n/<br\/>/g;
              $content .= $c;
            }
          }
        }
      }
      $content .= "<br/>-----------</p>\n";
    }
  }
  $content;
}
{$content}
```

C'est moche et long, je sais. Dites-vous que j'ai passé plus d'une après-midi pour trouver ça, la documentation est inexistante pour faire ça.

Les intervenants n'auront plus qu'à copier-coller l'article qui se trouve au bas de leur mail de notification dans leur réponse :-)

Commandes par mail

C'est beau de répondre par mail, mais il faut encore se connecter à l'interface web pour effectuer certaines actions. Comme je suis fier d'être fainéant, j'ai créé un *scrip* pour autoriser certaines actions par mail.

Mais avant ça, précisions :

- RT permet aux intervenants de discuter du ticket *sans que cela soit vu par le créateur du ticket* : c'est le but de l'adresse `rt-comment@example.org` du début de l'article. On va utiliser cette adresse pour piloter RT par mail
- un *scrip* est une action effectuée par RT en réponse à un évènement, en utilisant de façon optionnelle un modèle. Typiquement, il y a un *scrip* qui envoie (action) un mail (d'après un modèle) aux membres du support lorsqu'un ticket est créé (évènement).

Créons donc un *scrip*. Menu `Administration > Scrips > Ajouter`.

- Condition (évènement) => Lors d'un commentaire
- Action => définie par l'utilisateur
- Modèle => Modèle vide

Dans le `Programme de préparation d'action personnalisé:` :

```
if ($self->TransactionObj->Attachments->First->Content =~
m/#(JePrends|Fermeture|Spam|Move:.*)/i) {
    return 1;
} else {
    return 0;
}
```

Oui, j'aurais pu faire un *one-liner*, mais il faut que ça reste lisible facilement, et quand on passe des heures à faire des bidouilles comme ça, on apprécie les codes lisibles en un coup d'œil.

Dans le `Code d'action personnalisée (commit):` :

```
if ($self->TransactionObj->Attachments->First->Content =~ m/#JePrends/i) {
    if ( $self->TicketObj->OwnerAsString eq '' ) {
        my $id = $self->TransactionObj->Creator;
        $RT::Logger->info("Setting owner to ".$id);
```

```
        $self->TicketObj->SetOwner($id, 'SET');
    }
} elsif ($self->TransactionObj->Attachments->First->Content =~ m/#Fermeture/i) {
    $RT::Logger->info("Closing ticket");
    $self->TicketObj->SetStatus('resolved');
} elsif ($self->TransactionObj->Attachments->First->Content =~ m/#Spam/i) {
    my $ticket = $self->TicketObj;
    my ($status, $msg) = $ticket->SetStatus('rejected');
    $RT::Logger->error("Couldn't delete ticket: $msg") unless $status;

    my $requestors = $ticket->Requestor->UserMembersObj;
    while (my $requestor = $requestors->Next) {
        $requestor->SetDisabled(1);
        $RT::Logger->info("Disabling user ".$requestor->Format." because he's likely a
spammer");
    }
} elsif ($self->TransactionObj->Attachments->First->Content =~ m/#Move:(.*)/i) {
    my $new_queue = $1;
    my $ticket    = $self->TicketObj;

    my ($result, $msg) = $ticket->SetQueue($new_queue);
    if ($result) {
        $RT::Logger->info("Moving ticket to queue ".$new_queue);
    } else {
        $RT::Logger->error("Error while moving ticket to queue ".$new_queue.": ".$msg);
    }
} elsif ($self->TransactionObj->Attachments->First->Content =~ m/#Merge:(.*)/i) {
    my $target = $1;
    my $ticket = $self->TicketObj;

    $ticket->MergeInto($target);
    $RT::Logger->info("Merging ticket into ticket ".$target);
}

return 1;
```

Voilà, enregistrez et c'est bon.

Lorsqu'un commentaire contiendra une commande, elle sera exécutée :

- `#JePrends` => l'intervenant s'assigne le ticket
- `#Fermeture` => le ticket est marqué comme résolu
- `#Spam` => le ticket est supprimé et son auteur ne pourra plus ouvrir de tickets, son adresse mail sera blacklistée
- `#Move:file_de_tickets` => le ticket est basculé vers la file de tickets spécifiée
- `#Merge:no_ticket` => fusionne le ticket avec le ticket dont on donne le n°

Et le spam alors ?

Pour le spam, préparez d'abord un `spamassassin` pour votre serveur de mails. Ce n'est pas l'objet de cet article, il n'y a qu'à fouiller un peu le web pour trouver des tutos.

On va recréer un *scrip*, mais avant cela on va créer une nouvelle file nommée `spam` (menu `Administration > Files > Ajouter`).

Pour notre nouveau *scrip* :

- Condition (évènement) => Lors d'une création
- Action => définie par l'utilisateur
- Modèle => Modèle vide

Dans le `Programme de préparation d'action personnalisé:` :

```
if ( $self->TicketObj->Subject !~ /\[ .* \]/i ) {
  my $inMessage = $self->TransactionObj->Attachments->First;

  # if no message attachment - assume web UI
  return 0 if (!$inMessage);

  # exit if not email message
  return 0 if (!$inMessage->GetHeader('Received'));

  return ($inMessage->GetHeader('X-Spam-Level') =~ m/\*+/) ? 1 : 0;
} else {
  return 1;
}
```

Dans le `Code d'action personnalisée (commit):` :

```
my $spamlevel = $self->TransactionObj->Attachments->First->GetHeader('X-Spam-Level');
if ($spamlevel =~ m/\*\*\*+/) {
```

```
  if ($spamlevel =~ m/\*\*\*\*\*/) {
    $RT::Logger->info("This mail seems to be a spam => deleting");
    $self->TicketObj->Delete();
  } else {
    $RT::Logger->info("This mail seems to be a spam => queue spam");
    $self->TicketObj->SetQueue('spam');
  }
}
return 1;
```

Avec cela, les mails ayant un score de 5 ou plus au *spamLevel* seront supprimés, et ceux qui ont entre 3 et 5 vont au purgatoire, dans la file `spam`.

Prenez soin de déplacer ce *scrip* tout en haut de la liste pour qu'il soit le premier exécuté.

Plugins

En vrac, les plugins que j'utilise :

- RT::Authen::ExternalAuth::LDAP
- RT::Extension::LDAPImport
- RT::Extension::ReportSpam

Les deux premiers sont maintenant intégrés à RT, il n'y a pas besoin de les installer, juste de les configurer. Ils servent respectivement à assurer l'authentification LDAP à l'interface web, et à importer en masse les comptes du LDAP pour permettre à l'administrateur de mettre les collaborateurs dans les bons groupes sans attendre qu'ils se soient logués une première fois.

Le dernier plugin ajoute un `S` dans le menu des tickets, permettant de les déclarer comme spam d'un simple clic.

Conclusion

On peut faire de merveilleuses choses avec RT, pour peu que l'on ait le temps de fouiller dans la documentation officielle... et dans le code !

Une fois bien configuré, il devrait permettre d'alléger la charge de travail du groupe de support et je peux vous dire que pour en faire depuis plus de deux ans pour Framasoft et bien plus pour mon ancien boulot, ce n'est pas quelque chose à négliger □□

NB : bien évidemment, ce superbe logiciel est en Perl :D

TinyTinyRSS : rétablir la consultation web des articles publiés (+ bonus)

On peut, dans TTRSS, publier des articles de ses flux RSS. Cela permet de partager aisément des articles. Les articles sont publiés dans un flux RSS que d'autres personnes pourront à leur tour mettre dans un lecteur de flux RSS.

Je couple cette fonctionnalité avec Feed2toot, qui me permet d'envoyer un pouet sur Mastodon pour chaque nouvel article que je publie.

Le flux RSS généré a longtemps eu un document XSLT associé afin de rendre le flux lisible dans un navigateur web. Ce document XSLT a été supprimé le 28 mars 2020 par le projet TTRSS, mais il est fort simple de le remettre en place.

Ajouter la référence au document XSLT

On va commencer par copier le fichier modèle du flux dans le dossier `templates.local` : cela permettra de conserver les modifications de ce fichier malgré les mises à jour de TTRSS.

```
cp templates/generated_feed.txt templates.local/generated_feed.txt
```

On ajoute alors cette ligne à la ligne 2 du fichier copié :

```
<?xml-stylesheet type="text/xsl" href="templates.local/atom-to-html.xsl"?>
```

Vous noterez que je place le document XSLT dans le dossier `templates.local` : il me semble logique de mettre mes modifications dans un dossier `*.local`.

Créer le document XSLT

Je suis parti du fichier précédemment fourni par TTRSS, mais je l'ai un peu modifié pour :

- ajouter une feuille de style qui permet d'avoir un thème sombre pour les visiteurs qui utilisent un thème sombre sur leur système d'exploitation (voir la caractéristique média `prefers-color-scheme`. Attention, ça ne fonctionne pas sur tous les navigateurs et tous les système d'exploitation.
- ajouter une ancre sur les titres des articles, me permettant ainsi de mettre un lien qui enverra les visiteurs au bon article, quand bien même d'autres ont été publiés depuis (tant que l'article est dans le flux RSS, bien sûr, les flux RSS ne contenant qu'un nombre fini d'articles)

Voici mon fichier `templates.local/atom-to-html.xsl` :

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
  xmlns:atom="http://www.w3.org/2005/Atom"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">

<xsl:output method="html"/>

<xsl:template match="/atom:feed">
<html>
  <head>
    <title><xsl:value-of select="atom:title"/></title>
    <link rel="stylesheet" type="text/css" href="themes/light.css"/>
    <link rel="stylesheet" type="text/css" href="themes.local/dark-mode.css"/>
    <script language="javascript" src="lib/xsl_mop-up.js"></script>
  </head>

  <body onload="go_decoding()" class="ttrss_utility">

    <div id="cometestme" style="display:none;">
      <xsl:text disable-output-escaping="yes">&amp;amp;</xsl:text>
    </div>

    <div class="rss">

    <h1><xsl:value-of select="atom:title"/></h1>
```

```
    <p class="description">This feed has been exported from
      <a target="_new" class="extlink" href="http://tt-rss.org">Tiny Tiny RSS</a>.
      It contains the following items:</p>

    <xsl:for-each select="atom:entry">
      <h2 id="{atom:id}"><a target="_new" href="{atom:link/@href}"><xsl:value-of
select="atom:title"/></a></h2>

      <div name="decodeme" class="content">
        <xsl:value-of select="atom:content" disable-output-escaping="yes"/>
      </div>

      <xsl:if test="enclosure">
        <p><a href="{enclosure/@url}">Extra...</a></p>
      </xsl:if>


    </xsl:for-each>

    </div>

  </body>
 </html>
</xsl:template>

</xsl:stylesheet>
```

Et voici la feuille de style pour le thème sombre automatique, que je place dans `themes.local/dark-mode.css` :

```
@media (prefers-color-scheme: dark) {
    * {
        scrollbar-color: #2a2c2e #1c1e1f;
    }
    html, body, input, textarea, select, button {
        background-color: #181a1b;
    }
    html, body, input, textarea, select, button {
```

```
        border-color: #575757;
        color: #e8e6e3;
    }
    body.ttrss_utility {
        background-image: initial;
        background-color: rgb(27, 29, 30);
        color: rgb(232, 230, 227);
    }
    body.ttrss_utility .content {
        background-image: initial;
        background-color: rgb(24, 26, 27);
        border-top-color: rgb(58, 58, 58);
        border-right-color: rgb(58, 58, 58);
        border-bottom-color: rgb(58, 58, 58);
        border-left-color: rgb(58, 58, 58);
        box-shadow: rgba(0, 0, 0, 0.1) 0px 1px 1px -1px;
    }
}
```

Configuration Feed2toot

Pour l'installation de Feed2toot, je vous laisse regarder la documentation officielle, ce n'est pas l'objet de cet article.

Voici ma configuration Feed2toot, qui poste le lien vers mon flux, le titre de l'article, l'adresse d'origine de l'article et le résumé de l'article :

```
[mastodon]
instance_url=https://framapiaf.org
user_credentials=/etc/feed2toot/feed2toot_usercred.txt
client_credentials=/etc/feed2toot/feed2toot_clientcred.txt

[cache]
cachefile=/opt/feed2toot.db

[rss]
uri=https://ttrss.fiat-tux.fr/public.php?op=rss&id=-
2&key=60c63a21c2928546b4485017876fe850c6ebcebd
```

```
toot=[veille] https://lstu.fr/veille-luc#{id} « {title} » {link} {summary}
```

Notez le `#{id}` après `https://lstu.fr/veille-luc` (qui est une URL raccourcie vers mon flux RSS) : cela permet de renvoyer pile à l'article plutôt que de laisser les gens descendre et rechercher l'article □□

Une web radio avec MPD

Rien de plus simple que de faire une web radio pour diffuser sa musique.

MPD

On installe MPD :

```
apt install mdp
```

On modifie sa configuration dans `/etc/mpd.conf` (je ne mets que les paramètres intéressants à modifier) :

```
# chemin vers le dossier contenant la musique
music_directory "/var/lib/mpd/music"
# Là c’est vous qui voyez. Moi je mets à off pour pas que ça surveille le dossier en
permanence
# Je déclenche la mise à jour de la bibliothèque via le client MPD
auto_update "no"
# Zeroconf/avahi, c’est intéressant dans un réseau local, beaucoup moins sur un serveur
zeroconf_enabled "no"
# On commente l’`audio_output` de type `alsa`
# et on décommente celui de type `httpd`
audio_output {
       type            "httpd"
       name            "Ma radio perso"
 #      encoder          "vorbis"                # optional, vorbis or lame
       port            "8000"
       bind_to_address "127.0.0.1"              # optional, IPv4 or IPv6
 #      quality          "5.0"                  # do not define if bitrate is defined
       bitrate         "128"                   # do not define if quality is defined
 #      format           "44100:16:1"
 #      max_clients      "0"                     # optional 0=no limit
}
```

On redémarre MPD :

```
systemctl restart mpd
```

Un client

J'ai tendance à préférer les outils en CLI :

```
apt install ncmpcpp
```

Lancez-le.

- `F1` pour afficher l'aide.
- `u` pour mettre à jour la mise à jour de la bibliothèqie.
- `z` pour utiliser le mode aléatoire
- `r` pour metter la file d'attente en boucle
- `1` pour voir la file d'attente des fichiers. Dans cette vue :
  - `Entrée` pour lancer un morceau
- `2` pour voir la liste des fichiers (suivant l'arborescence de votre dossier de musique). Dans cette vue :
  - `Espace` pour ajouter à la file d'attente,
  - `Entrée` pour aller dans un dossier.

Pour le reste, lisez l'aide.

Proxifier via Nginx

Pas grand chose de particulier, c'est de la proxification classique :

```
    location / {
        proxy_set_header Host $host;
        proxy_connect_timeout 300s;
        proxy_read_timeout 300s;
        proxy_send_timeout 300s;
        proxy_pass http://127.0.0.1:8000/;
    }
```

(je mets pas toute la configuration Nginx, c'est pas le sujet ici)

Vaultwarden

Comment compiler et installer proprement le clone de Bitwarden en Rust. Les bases de données disponibles à ce jour sont PostgreSQL, MySQL et SQLite.

NB: Vaultwarden s'appelait précédemment Bitwarden_rs et a été renommé le 27 avril 2021. Cette page a été remaniée en conséquence. N'hésitez pas à me signaler des soucis (voir l'adresse mail sur ma page de présentation).

NB: J'ai supprimé les informations pour compiler soit-même l'interface web, il fallait NodeJs 14 minimum, ce qui implique de rajouter les explications pour installer cette version de NodeJs. Bref, flemme. Le tutoriel utilise maintenant les *releases* créées par Dani Garcia (auteur de Vaultwarden).

Compilation

Installation des dépendances :

```
sudo apt install pkg-config libssl-dev build-essential
```

Si vous voulez utiliser MySQL comme base de données :

```
sudo apt install default-libmysqlclient-dev
```

Si vous voulez utiliser PostgreSQL comme base de données :

```
sudo apt install libpq-dev
```

Installation de Rust

Installation de rustup, qui nous fournira le compilateur Rust :

```
curl https://sh.rustup.rs -sSf > rustup.sh
```

On n'exécute pas direct un script tiré du web ! On regarde d'abord s'il ne va pas faire de saloperies :

```
vi rustup.sh
```

On le rend exécutable :

```
chmod +x rustup.sh
```

On installe le compilateur Rust (il sera dans notre `$HOME`) :

```
./rustup.sh --default-host x86_64-unknown-linux-gnu --default-toolchain stable
```

> Attention ! Ceci n'est valable que pour l'architecture x86_64 ! Si vous voulez installer Vaultwarden sur une architecture ARM (comme les Raspberry Pi), il faut adapter la commande, a priori en remplaçant `x86_64-unknown-linux-gnu` par `armv7-unknown-linux-gnueabihf`.

On source un fichier qui nous permet de l'appeler

```
source $HOME/.cargo/env
```

Mise à jour de Rust (si vous l'avez déjà installé via rustup)

```
rustup update
```

Compilation

Clonez le projet vaultwarden.

```
git clone https://github.com/dani-garcia/vaultwarden
```

Compilation de Vaultwarden :

```
cd vaultwarden
# Pour les mises à jour
git fetch
git checkout -b "v$(git tag --sort=v:refname | tail -n1)" "$(git tag --sort=v:refname | tail -n1)"
# on supprime les artefacts de la compilation précédente
```

```
cargo clean
cargo build --release --features postgresql
# ou sqlite, ou mysql, selon la bdd que vous souhaitez utiliser
cd -
```

Le résultat de la compilation est dans `vaultwarden/target/release/`.

Récupération de l'interface web

D'abord, si vous ne l'avez pas déjà fait, récupérez la clé GPG de Dani Garcia et de Black Dex, un contributeur :

```
gpg --keyserver keyserver.ubuntu.com --recv-keys B9B7A108373276BF3C0406F9FC8A7D14C3CD543A \
        3C5BBC173D81186CFFDE72A958C80A2AA6C765E1 \
        13BB3A34C9E380258CE43D595CB150B31F6426BC
```

On va utiliser une variable pour le numéro de version, qu'on va aller chercher sur https://github.com/dani-garcia/bw_web_builds/releases, c'est plus simple pour les mises à jour (on change la variables, mais pas les commandes) :

```
VERSION=v2026.4.1
```

Ensuite, on récupère la version patchée de l'interface web avec `wget` :

```
wget -q --show-progress \
      "https://github.com/dani-
garcia/bw_web_builds/releases/download/$VERSION/bw_web_$VERSION.tar.gz" \
      "https://github.com/dani-
garcia/bw_web_builds/releases/download/$VERSION/bw_web_$VERSION.tar.gz.asc"
```

On vérifie la signature de l'archive:

```
gpg --verify "bw_web_$VERSION.tar.gz.asc" "bw_web_$VERSION.tar.gz"
```

On décompresse l'archive :

```
tar xvf "bw_web_$VERSION.tar.gz"
```

Si vous faites une mise à jour, supprimez l'ancienne version de l'interface web :

```
rm -rf vaultwarden/target/release/web-vault/
```

Et on déplace l'interface web dans le dossier où attend le résultat de la compilation de vaultwarden :

```
mv web-vault/ vaultwarden/target/release/web-vault/
```

Si vous faites uniquement une mise à jour de l'interface :

```
sudo rm -rf /opt/vaultwarden/web-vault/ &&
  sudo rsync -a --info=progress2 vaultwarden/target/release/web-vault/  /opt/vaultwarden/web-vault/ &&
  sudo chown -R www-data: /opt/vaultwarden/web-vault/
```

Pour une mise à jour

Suivez le tuto d'installation avec ces précautions préalables :

- coupez le service vaultwarden ;
- faites des sauvegardes de votre installation (fichiers, données de la base de données) avant de faire le `rsync` d'installation (voir plus bas). Pour les fichiers :

  ```
  sudo rsync -a --info=progress2 /opt/vaultwarden/ /opt/vaultwarden_$(date +%F).bak/
  ```

Installation

On va installer Vaultwarden dans `/opt/vaultwarden` et on le fera tourner avec l'utilisateur `www-data` :

```
sudo rm -rf /opt/vaultwarden/deps/* /opt/vaultwarden/build/*
sudo rsync -a --info=progress2 vaultwarden/target/release/ /opt/vaultwarden/
sudo chown -R www-data: /opt/vaultwarden
```

Puis on va créer un service `systemd`, `/etc/systemd/system/vaultwarden.service` :

```
[Unit]
Description=Vaultwarden Server (Rust Edition)
Documentation=https://github.com/dani-garcia/vaultwarden
After=network.target
```

```
[Service]
# The user/group vaultwarden is run under. the working directory (see below) should allow write and read access to this user/group
User=www-data
Group=www-data
# The location of the .env file for configuration
EnvironmentFile=/etc/vaultwarden.env
# The location of the compiled binary
ExecStart=/opt/vaultwarden/vaultwarden
# Set reasonable connection and process limits
LimitNOFILE=1048576
LimitNPROC=64
# Isolate vaultwarden from the rest of the system
PrivateTmp=true
PrivateDevices=true
ProtectHome=true
ProtectSystem=strict
# Only allow writes to the following directory and set it to the working directory (user and password data are stored here)
WorkingDirectory=/opt/vaultwarden/
ReadWriteDirectories=/opt/vaultwarden/

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

Pour l'interface d'administration, on va créer un token avec :

```
/opt/vaultwarden/vaultwarden hash
```

La configuration se fait via des variables d'environnement qu'on va mettre dans `/etc/vaultwarden.env` :

```
SIGNUPS_ALLOWED=false
WEBSOCKET_ENABLED=true
ADMIN_TOKEN=Un token généré avec `/opt/vaultwarden/vaultwarden hash`
ROCKET_ADDRESS=127.0.0.1
WEBSOCKET_ADDRESS=127.0.0.1
SMTP_HOST=127.0.0.1
SMTP_FROM=vaultwarden@example.org
SMTP_PORT=25
```

```
SMTP_SSL=false
```

Vous remarquerez que je dis à Vaultwarden d'envoyer les mails via le serveur SMTP local. À vous de faire en sorte qu'il fonctionne. Allez voir le wiki du projet ou le modèle de fichier d'environnement pour voir quelles variables vous pourriez ajouter, enlever, modifier… Vous pouvez faire ça pour voir les nouveautés :

```
sudo vimdiff -c 'map <F2> :diffget<cr>]czz | map <F3> ]czz | syn off | windo set wrap | winc
h' \
  /etc/vaultwarden.env vaultwarden/.env.template
```

Puis :

```
sudo systemctl daemon-reload
sudo systemctl enable --now vaultwarden
sudo systemctl status vaultwarden
```

Nginx

On installe Nginx s'il n'est pas déjà installé :

```
sudo apt install nginx
```

Configuration du virtualhost :

```
# The `upstream` directives ensure that you have a http/1.1 connection
# This enables the keepalive option and better performance
#
# Define the server IP and ports here.
upstream vaultwarden-default {
  zone vaultwarden-default 64k;
  server 127.0.0.1:8080;
  keepalive 2;
}

# Needed to support websocket connections
# See: https://nginx.org/en/docs/http/websocket.html
# Instead of "close" as stated in the above link we send an empty value.
# Else all keepalive connections will not work.
```

```
map $http_upgrade $connection_upgrade {
    default upgrade;
    ''      "";
}

server {
    listen 80;
    listen [::]:80;
    listen 443 ssl http2;
    listen [::]:443 ssl http2;
    server_name vaultwarden.example.org;

    access_log /var/log/nginx/vaultwarden.access.log;
    error_log /var/log/nginx/vaultwarden.error.log;

    ssl_certificate      /etc/letsencrypt/live/vaultwarden.example.org/fullchain.pem;
    ssl_certificate_key  /etc/letsencrypt/live/vaultwarden.example.org/privkey.pem;

    ssl_session_timeout 5m;
    ssl_session_cache shared:SSL:5m;

    ssl_prefer_server_ciphers On;
    ssl_protocols TLSv1.2;
    ssl_ciphers
'EECDH+aRSA+AESGCM:EECDH+aRSA+SHA384:EECDH+aRSA+SHA256:EECDH:+CAMELLIA256:+AES256:+CAMELLIA128
:+AES128:+SSLv3:!aNULL:!eNULL:!LOW:!3DES:!MD5:!EXP:!PSK:!DSS:!RC4:!SEED:!ECDSA';

    ssl_dhparam /etc/ssl/private/dhparam4096.pem;
    add_header Strict-Transport-Security max-age=15768000; # six months
    gzip off;

    # Redirect HTTP to HTTPS
    if ($https != 'on') {
        rewrite ^/(.*)$ https://vaultwarden.example.org/$1 permanent;
    }

    root /var/www/html;

    # Allow large attachments
    client_max_body_size 525M;
```

```
    location ^~ '/.well-known/acme-challenge' {
        default_type "text/plain";
        root /var/www/certbot;
    }

    location / {
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection $connection_upgrade;

        include /etc/nginx/proxy_params;
        ## /etc/nginx/proxy_params contient normalement ceci :
        #proxy_set_header Host $http_host;
        #proxy_set_header X-Real-IP $remote_addr;
        #proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        #proxy_set_header X-Forwarded-Proto $scheme;

        proxy_pass http://vaultwarden-default;
    }
}
```

Pour créer `/etc/ssl/private/dhparam4096.pem` :

```
sudo openssl dhparam -out /etc/ssl/private/dhparam4096.pem 4096
```

Pour le certificat Let’s Encrypt, on commente le brol relatif à `ssl` puis :

```
sudo nginx -t && sudo nginx -s reload
sudo apt install certbot
sudo mkdir /var/www/certbot/
certbot certonly --rsa-key-size 4096 --webroot -w /var/www/certbot/ --agree-tos --text --
renew-hook "/usr/sbin/nginx -s reload" -d vaultwarden.example.org
```

Une fois qu’on a le certificat, on décommente le brol `ssl` puis :

```
sudo nginx -t && sudo nginx -s reload
```

Sauvegarde

Créer le script de sauvegarde `/opt/backup_vaultwarden.sh` :

```
#!/bin/bash
function vwbackup {
    DATE=$(date '+%a%H')

    # Database, ONLY FOR SQLITE!
    if [[ ! -d /opt/backup_vaultwarden/sqlite-backup/ ]]; then
        mkdir -p /opt/backup_vaultwarden/sqlite-backup/
    fi
    echo ".backup /opt/backup_vaultwarden/sqlite-backup/db.${DATE}.sqlite3" | sqlite3
/opt/vaultwarden/data/db.sqlite3 2>> /opt/backup_vaultwarden/backup.log
    if [[ "$?" -ne "0" ]]; then
        echo "Something went wrong with Vaultwarden database backup, please see
/opt/backup_vaultwarden/backup.log on "$(hostname) | mail -s "Vaultwarden database backup"
youraddress@mail.example.org
        vwbackup
    fi

    # Files
    if [[ ! -d /opt/backup_vaultwarden/files-backup/ ]]; then
        mkdir -p /opt/backup_vaultwarden/files-backup/
    fi
    rsync -a --delete --exclude db.sqlite3 /opt/vaultwarden/data/
/opt/backup_vaultwarden/files-backup/$DATE/ 2>> /opt/backup_vaultwarden/backup.log
    if [[ "$?" -ne "0" ]]; then
        echo "Something went wrong with Vaultwarden files backup, please see
/opt/backup_vaultwarden/backup.log on "$(hostname) | mail -s "Vaultwarden files backup"
youraddress@mail.example.org
        vwbackup
    fi
}
vwbackup
```

Puis :

```
sudo chmod +x /opt/backup_vaultwarden.sh
sudo mkdir /opt/backup_vaultwarden
sudo chown www-data: /opt/backup_vaultwarden
sudo apt install sqlite3 ## Si vous utilisez SQLite
```

Puis, dans le cron de l'utilisateur `www-data` :

```
42 4 * * * /opt/backup_vaultwarden.sh
```

Logs

J'aime bien avoir mes logs dans un dossier dédié pour ce genre de service.

Dans `/etc/rsyslog.d/vaultwarden.conf` :

```
if $programname == 'vaultwarden' then /var/log/vaultwarden/vaultwarden.log
if $programname == 'vaultwarden' then ~
```

Dans `/etc/logrotate.d/vaultwarden` :

```
/var/log/vaultwarden/vaultwarden.log
{
        rotate 52
        dateext
        weekly
        missingok
        notifempty
        compress
        sharedscripts
        postrotate
                invoke-rc.d rsyslog rotate > /dev/null
        endscript
}
```

Puis :

```
sudo mkdir /var/log/vaultwarden
sudo chown root:adm /var/log/vaultwarden
sudo systemctl restart rsyslog
```bash

## Fail2ban

Un fail2ban qui surveille les logs, ça permet de bloquer les petits malins qui font du
```
```

```
bruteforce

```bash
sudo apt install fail2ban
```

Dans `/etc/fail2ban/filter.d/vaultwarden.conf` :

```
[INCLUDES]
before = common.conf

[Definition]
failregex = ^.*Username or password is incorrect\. Try again\. IP: <HOST>\. Username:.*$
ignoreregex =
```

Dans `/etc/fail2ban/jail.d/vaultwarden.local` :

```
[vaultwarden]
enabled = true
port = 80,443
filter = vaultwarden
action = iptables-allports[name=vaultwarden]
logpath = /var/log/vaultwarden/vaultwarden.log
maxretry = 3
bantime = 14400
findtime = 14400
```

Pour la page d'admin, dans `/etc/fail2ban/filter.d/vaultwarden-admin.conf` :

```
[INCLUDES]
before = common.conf

[Definition]
failregex = ^.*Unauthorized Error: Invalid admin token\. IP: <HOST>.*$
ignoreregex =
```

Dans `/etc/fail2ban/jail.d/vaultwarden-admin.local` :

```
[vaultwarden-admin]
enabled = true
port = 80,443
filter = vaultwarden-admin
```
```

```
action = iptables-allports[name=vaultwarden]
logpath = /var/log/vaultwarden/vaultwarden.log
maxretry = 3
bantime = 14400
findtime = 14400
```

Finalement :

```
sudo systemctl restart fail2ban
```

Conclusion

Voilà, vous devriez avoir un serveur Vaultwarden fonctionnel. Plus qu'à aller sur l'interface web que vous venez de mettre en place ou télécharger les clients et à les utiliser !

Pour importer vos mots de passe de Firefox, il faut passer par une application pour les exporter, puis aller dans les outils de votre client (ou de l'interface web).

Wisemapping

Installation de quelques outils dont on a besoin

```
sudo apt install jetty9 maven git nodejs npm
```

Récupération des sources

```
git clone https://github.com/wisemapping/wisemapping-open-source.git
```

Configuration et compilation

```
cd wisemapping-open-source
# On configure
vi wise-webapp/src/main/webapp/WEB-INF/app.properties
mvn package
```

Pour virer les appels à *Google analytics*, après la compilation :

```
for i in $(grep -Rcl google-analytics)
do
    sed -i -e "s@https://www\.google-analytics\.com/analytics\(_debug\)\?\.js@@g" $i
done
for i in $(grep -Rcl googletagmanager); do
    sed -i -e "s@https://www\.googletagmanager\.com/@#@g" $i
done
mvn package
```

Pour créer et initialiser la base de données, regarder dans `config/database/`, modifier les fichiers (parce que le mot de passe `password`, c'est très bof, et ça force le nom de la base de données utilisée) et exécuter le SQL.

Installation dans Jetty

```
cp -a wise-webapp/target/wisemapping.war /usr/share/jetty9/webapps/root.war
chown jetty: /usr/share/jetty9/webapps/root.war
systemctl restart jetty9.service
```

Jetty devrait écouter de base sur `http://127.0.0.1:8080` (il me semble), y a plus qu'à mettre un nginx devant avec, pour l'essentiel :

```
    location / {
        include proxy_params;
        proxy_pass  http://127.0.0.1:8080;
    }
```

Pour que Jetty prenne en compte les en-têtes `X-Forwarded` envoyés par Nginx (configurés dans `/etc/nginx/proxy_params`), il faut ajouter `http-forwarded` à la liste des modules de Jetty, dans `/etc/jetty9/start.ini`. Par exemple, il faut passer de

```
--module=deploy,http,jsp,jstl,websocket,ext,resources
```

à

```
--module=deploy,http,jsp,jstl,websocket,ext,resources,http-forwarded
```

et redémarrer Jetty :

```
systemctl restart jetty9.service
```

Cela permet à Jetty de rediriger vers la bonne adresse (sans ça, à la connexion, il renvoie vers la version non-sécurisée (`http`) du site quand bien même on a pris soin de mettre en place du `https`.