Création d'un serveur de messagerie dans un conteneur

Table of Contents

De nombreux projets nécessitent la possibilité d'envoyer des emails, mais la création d'un serveur de messagerie peut aussi être un projet à part entière.

Cet article a pour objet d'expliquer la création d'un serveur autohébergé dans un conteneur **LXC**

Création du conteneur

Créer un nouveau conteneur dans lequel exécuter le serveur de messagerie :

```
apt-get -y installer lxc1
lxc-create -t télécharger -n mail -- -d ubuntu -r bionic -a amd64
```

Lui donner une adresse IP statique via dnsmasq :

```
echo "dhcp-host=mail,10.0.5.155" >> /etc/lxc/dnsmasq.conf
echo "LXC_DHCP_CONFILE=/etc/lxc/dnsmasq.conf" >> /etc/default/lxc-net
sudo systemctl stop lxc-net
sudo systemctl start lxc-net
```

Le but de l'adresse IP statique est de faciliter le transfert des ports liés au services hébergés dans le

conteneur. J'ai fait cela avec un script lancé au démarrage par systemd :

```
cat > /usr/bin/container-ports-fwd << EOF
nic=enp0s31f6
iptables -t nat -A PREROUTING -p tcp -i ${nic} --dport 25 -j DNAT --to-
destination 10.0.5.155:25
iptables -t nat -A PREROUTING -p tcp -i ${nic} --dport 465 -j DNAT --to-
destination 10.0.5.155:465
iptables -t nat -A PREROUTING -p tcp -i ${nic} --dport 993 -j DNAT --to-
destination 10.0.5.155:993
iptables -t nat -A PREROUTING -p tcp -i ${nic} --dport 587 -j DNAT --to-
destination 10.0.5.155:587
EOF

chmod 755 /usr/bin/container-ports-fwd

cat > /etc/systemd/system/container-ports-forward.service << EOF
[Unit]
Description=Bring up port forwards for lxc
After=lxc-net.target
Before=lxc.service

[Service]
Type=oneshot
RemainAfterExit=yes
ExecStart=/usr/bin/container-ports-fwd

[Install]
WantedBy=multi-user.target
EOF

systemctl daemon-reload
systemctl enable container-ports-fwd
systemctl start container-ports-fwd
```

Installer le Serveur Mail

Cette section va montrer les différentes étapes de la mise en place d’un serveur mail, que ce soit un simple SMTP ou une suite complète avec webmail inclus.

Postfix est le service principal à installer pour héberger un serveur de messagerie. Il permet d’envoyer et recevoir des e-mails. D’autres services peuvent ensuite être ajoutés, comme Dovecot pour le support POP/IMAP et Roundcube pour la création d’un webmail.

Il faut modifier les enregistrements MX du fournisseur de nom de domaine et modifier ces zones pour qu’elles correspondent à l'adresse IP actuelle du serveur pour recevoir des e-mails.

Installer Postfix pour envoyer des e-mails

Postfix est la base du serveur de messagerie. Il permettra d'envoyer et de recevoir des emails correspondant au nom de domaine.

```
sudo apt install postfix
```

Lors de l'installation, il faut choisir ces deux options de configuration :

- Le type général de configuration du courrier : Site Internet
- Nom de messagerie du système : domain.com

Maintenant effectuer deux changements dans la configuration qui a été générée :

Ouvrir le fichier de configuration :

```
sudo vi /etc/postfix/main.cf
```

Désactiver la gestion IPv6 :

- Remplacer : `inet_protocols = all`
- Par : `inet_protocols = ipv4`

Entrer le nom de domaine dans `myhostname`:

```
myhostname= domaine.com
```

Si le fournisseur d'accès à Internet ne permet pas d'envoyer des e-mails directement depuis un réseau local. Il faut demander à votre fournisseur le serveur à utiliser comme relais pour l'ajouter dans la configuration.

`relayhost = smtp.votrefournisseur.com`

Sauvegarder, quitter puis redémarrer **Postfix**:

```
sudo service postfix restart
```

À ce stade, le serveur devrait démarrer correctement sans erreurs.

Maintenant faire un premier test en envoyant un email depuis le Raspberry Pi.

Pour ce test, on va utiliser **telnet** pour se connecter à postfix.

Installer **telnet**:

```
sudo apt-get install telnet
```

Se connecter au serveur SMTP :

```
telnet localhost 25
```

Entrer cette série de commandes :

```
ehlo
mail from: you@domaine.com
rcpt to: user@mail.com
data
Subject: test
Test
.
quit
```

Cette séquence de commandes va créer un e-mail et l'envoyer à utilisateur@mail.com(adresse e-mail externe).

Voici la trace complète :

```
pi@raspberrypi:~ $ telnet localhost 25
Trying ::1...
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 domaine.com ESMTP Postfix (Raspbian)
ehlo domaine.com
250-domaine.com
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250-STARTTLS
250-ENHANCEDSTATUSCODES
250-8BITMIME
250-DSN
250 SMTPUTF8
mail from: moi@domaine.com
250 2.1.0 Ok
rcpt to: destinataire@gmail.com
250 2.1.5 Ok
data
354 End data with <CR><LF>.<CR><LF>
Subject: test
Test
.
```

```
250 2.0.0 Ok: queued as 44EAE1FE54
quit
221 2.0.0 Bye
```

Pour le faire avec un moyen plus convivial, on peut installer **mailutils** pour utiliser la commande **mail**:
n
- Installer mailutils : `sudo apt install mailutils`
- Envoyer un courriel de test avec la commande mail : `echo 'Test' | mail -s "Test commande mail" destinataire@gmail.com`

Mutt est un client de messagerie libre en mode console pour les systèmes UNIX.

Dans tous les cas, on peut consulter le fichier journal `/var/log/mail.log` pour voir ce qui se passe si on ne reçoit pas l'e-mail.
\\Si tout fonctionne correctement, on doit voir quelque chose comme ceci :
`Jul 1 04:14:32 raspberrypi postfix/local[5433] : 734AA1FF96 : to=, relay=local, delay=0.09, delays=0.01/0.04/0/0.04, dsn=2.0.0, status=sent (delivered to mailbox)`

Recevoir des e-mails avec Postfix

Il est maintenant temps d'éditer notre configuration **Postfix** pour recevoir des emails.

Pour ce faire, on utilisera le format de boîtes aux lettres **Maildir**.

Maildir est un moyen sûr et facile de stocker des courriels : chaque boîte aux lettres est un répertoire, et chaque courriel est un fichier.

Modifier le fichier de configuration :

```
sudo vi /etc/postfix/main.cf
```

Ajouter ces lignes à la fin du fichier :

```
home_mailbox = Maildir/
```

```
mailbox_command =
```

Cette configuration indiquera à **Postfix** de créer un dossier `Maildir` pour chaque utilisateur du système. Ce dossier accueillera désormais les nouveaux courriels entrants.

Maintenant, créer le modèle de dossier `Maildir` en suivant ces étapes :

Installer ces paquets :

```
sudo apt install dovecot-common dovecot-imapd
```

Créer des dossiers dans le répertoire des modèles :

```
sudo maildirmake.dovecot /etc/skel/Maildir
sudo maildirmake.dovecot /etc/skel/Maildir/.Drafts
sudo maildirmake.dovecot /etc/skel/Maildir/.Sent
sudo maildirmake.dovecot /etc/skel/Maildir/.Spam
sudo maildirmake.dovecot /etc/skel/Maildir/.Trash
sudo maildirmake.dovecot /etc/skel/Maildir/.Templates
```

Ces modèles seront utilisés lorsqu'on ajoutera de nouveaux utilisateurs, mais pour ceux qui existent déjà, il faut le faire manuellement.

Par exemple, exécuter ces commandes pour pi :

```
sudo cp -r /etc/skel/Maildir /home/pi/
sudo chown -R pi:pi /home/pi/Maildir
sudo chmod -R 700 /home/pi/Maildir
```

Maintenant répéter le même type de test que précédemment, mais en mettant l'utilisateur pi en destinataire :

```
echo "Test" | mail -s "Test" pi@domaine.com
```

Et ensuite, vérifier que le courrier est arrivé dans le dossier Maildir :

```
pi@raspberrypi:~ $ cat
/home/pi/Maildir/new/1625109614.Vb302I205f1M127492.raspberrypi
 Return-Path:
 X-Original-To: pi@rpi.tips
 Delivered-To: pi@rpi.tips
 Received: by rpi.tips (Postfix, from userid 1000)
id 1CC5A205F2; Thu,  1 Jul 2021 04:20:14 +0100 (BST)
 Subject: Test commande mail
 To: pi@rpi.tips
 X-Mailer: mail (GNU Mailutils 3.5)
 Message-Id: 20210701032014.1CC5A205F2@rpi.tips
```

```
Date: Thu,  1 Jul 2021 04:20:14 +0100 (BST)
From: pi@raspberrypi
Test
```

Comme il ne devrait y avoir qu'un seul courrier dans le nouveau dossier, on peut utiliser l'autocomplétion (touche tabulation) pour le trouver.

Comme on peut le voir, l'adresse du chemin de retour n'est pas correcte, il faut changer le nom du serveur pour corriger cela :

```
sudo hostname domain.com
```

Installer Dovecot pour permettre les connexions POP et IMAP

Comme on a déjà installé Dovecot à l'étape précédente pour créer des dossiers Maildir. La seule chose qu'il reste à faire est de finaliser la configuration.

Ouvrir le fichier de configuration de Dovecot :

```
sudo vi /etc/dovecot/dovecot.conf
```

Supprimer le support IPV6 :

- Remplacer : `#listen = *, : :`
- Par : `listen = *`

Ouvrir le fichier de configuration du courrier de **Dovecot**:

```
sudo vi /etc/dovecot/conf.d/10-mail.conf
```

Editer le dossier Maildir :

- Remplacer : `mail_location = mbox:~/mail:INBOX=/var/mail/%u`
- Par : `mail_location = maildir:~/Maildir`

Ouvrir le fichier de configuration maître de **Dovecot**:

```
sudo vi /etc/dovecot/conf.d/10-master.conf
```

Indiquer à **Dovecot** d'écouter l'authentification SASL :

- Commenter toutes les lignes du paragraphe sur l'authentification du service par défaut (ajouter # avant chaque ligne).
- Ajouter ces lignes à la fin du fichier :

```
service auth { unix_listener /var/spool/postfix/private/auth { mode = 0660
user = postfix group = postfix } }
```

Ouvrir le fichier de configuration de l'authentification de **Dovecot**:

```
sudo vi /etc/dovecot/conf.d/10-auth.conf
```

Autoriser l'authentification en clair.

- Décommenter et modifier cette ligne : `#disable_plaintext_auth = yes`
- Pour devenir celui-ci : `disable_plaintext_auth = no`
- Modifier également cette ligne : `auth_mechanisms = plain login`

Modifier le fichier de configuration de **Postfix**:

```
sudo vi /etc/postfix/main.cf
```

Dire à Postfix d'utiliser SASL (ajouter ces lignes) :

```
smtpd_sasl_type = dovecot
smtpd_sasl_path = private/auth
smtpd_sasl_auth_enable = yes
Redémarrez Dovecot et Postfix :
sudo service postfix restart
sudo service dovecot restart
```

Pour vérifier que le service fonctionne correctement, garder un œil sur les fichiers journaux, et utiliser `sudo service X status`

C'est suffisant pour pouvoir envoyer et recevoir du courrier.

Sécurisation du serveur de messagerie

Il y a quelques options à mettre en place pour sécuriser un minimum le serveur web.

Blocage des accès

On peut modifier le fichier de configuration pour limiter l'utilisation du SMTP au réseau local et rejetter les personnes disant qu'elles proviennent du nom de domaine:

```
sudo vi /etc/postfix/main.cf
```

Ajouter ces lignes à la fin du fichier :

```
smtpd_helo_restrictions =
permit_mynetworks =
permit_sasl_authenticated =
reject_invalid_helo_hostname =
reject_non_fqdn_helo_hostname =
reject_unknown_helo_hostname =
check_helo_access =
hash:/etc/postfix/helo_access =
```

Créer le fichier helo_access :

```
sudo vi /etc/postfix/helo_access
```

Dans ce fichier, il faut mettre la liste des noms de domaine que l'on veut bloquer:

```
X.X.X.X REJECT
domaine.com REJECT
smtp.domaine.com REJECT
mail.domaine.com REJECT
```

 Remplacer X.X.X.X par l'adresse IP publique du serveur.

Redémarrer le service postfix :

```
sudo service postfix restart
```

Utilisaton de clès de chiffrement

Installer et exécuter **Letsencrypt** sur l'hôte (pas le conteneur):

```
sudo apt -y install letsencrypt
letsencrypt -d mail.example.org -m me@my.mail certonly
```

Copier les clés **Letsencrypt** dans le conteneur:

```
lxc-attach -n mail -- mkdir -p /etc/letsencrypt/live/mail.example.org
cp /etc/letsencrypt/live/mail.example.org/*
/var/lib/lxc/mail/rootfs//etc/letsencrypt/live/mail.example.org
```

Editer `/etc/postfix/main.cf` et `/etc/dovecot/conf.d/10-ssl.conf` pour pointer vers ceux qui utilisent ces lignes :

```
smtpd_tls_cert_file = /etc/letsencrypt/live/mail.example.org/fullchain.pem
smtpd_tls_key_file = /etc/letsencrypt/live/mail.example.org/privkey.pem
```

On a maintenant un serveur de messagerie fonctionnel et sécurisé. On va rendre ce serveur de messagerie accessible aux clients POP et IMAP via une authentification SASL.

Création d'un utilisateur

Pour vérifier que l'authentification SASL fonctionne bien, on va créer un utilisateur de test et essayer de se connecter au serveur de messagerie avec celui-ci.

```
sudo adduser test
```

Répondre aux questions relatives au mot de passe, les autres questions ne sont pas obligatoires (appuyer sur Entrée pour les ignorer).

Pour tester la connexion, il coder le mot de passe format en base64:

```
printf '\0%s\0%s' '[LOGIN]' '[MDP]' | openssl base64
```

Remplacer [LOGIN] et [MDP] dans cette commande par ceux choisis avec la commande adduser.

On peut maintenant réessayer une connexion avec telnet en spécifiant cette chaîne pour l'identification. Le seul changement est que l'on va utiliser la commande AUTH PLAIN pour se connecter :

```
telnet localhost 25
ehlo rpi.tips
AUTH PLAIN
```

On peut arrêter après cela si on a le message « Authentication successful », ou envoyer un autre email de test comme la première fois.

Voici la trace complète :

```
pi@raspberrypi:~ $ telnet localhost 25
Trying ::1...
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 domaine.com ESMTP Postfix (Raspbian)
ehlo domaine.com
250-domaine.com
```

```
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250-STARTTLS
250-AUTH PLAIN LOGIN
250-ENHANCEDSTATUSCODES
250-8BITMIME
250 DSN
AUTH PLAIN AHRlc3QAcGFzc3dvcmQ=
235 2.7.0 Authentication successful
mail from: moi@domaine.com
250 2.1.0 Ok
rcpt to: destinataire@gmail.com
250 2.1.5 Ok
data
354 End data with <CR><LF>.<CR><LF>
Subject: test
Test
.
250 2.0.0 Ok: queued as 44EAE1FE54
quit
221 2.0.0 Bye
```

Activer IMAPS

Dovecot permet de se connecter avec IMAP (telnet localhost 143), mais il faut activer TLS pour IMAP sur le port 993.

Editer le fichier de configuration du maître de **Dovecot** :

```
sudo vi /etc/dovecot/conf.d/10-master.conf
```

Activer le listener sur le port 993, la configuration devrait ressembler à ceci (il y a quelques lignes à décommenter) :

```
service imap-login {
inet_listener imap {
port = 143
}
inet_listener imaps {
port = 993
ssl = yes
}
}
```

Ensuite, éditer le fichier de configuration SSL :

```
sudo vi /etc/dovecot/conf.d/10-ssl.conf
```

S'assurer que SSL est activé au début du fichier :

```
ssl = yes
```

Les emplacements des certificats doivent également être décommentés :

```
ssl_cert = </etc/dovecot/private/dovecot.pem
ssl_key = </etc/dovecot/private/dovecot.pem
```

Enfin, redémarrer le serveur Dovecot :

```
sudo service dovecot restart
```

On peut maintenant vérifier que le serveur IMAPS fonctionne, avec cette commande :

```
openssl s_client -connect localhost:993
```

La syntaxe du login est la suivante :

```
a login [LOGIN] [MDP]
```

La trace complète devrait ressembler à ceci :

```
* OK [CAPABILITY IMAP4rev1 LITERAL+ SASL-IR LOGIN-REFERRALS ID ENABLE IDLE
AUTH=PLAIN AUTH=LOGIN] Dovecot ready.
a login pi mdp
a OK [CAPABILITY IMAP4rev1 LITERAL+ SASL-IR LOGIN-REFERRALS ID ENABLE IDLE
SORT SORT=DISPLAY THREAD=REFERENCES THREAD=REFS THREAD=ORDEREDSUBJECT
MULTIAPPEND URL-PARTIAL CATENATE UNSELECT CHILDREN NAMESPACE UIDPLUS LIST-
EXTENDED I18NLEVEL=1 CONDSTORE QRESYNC ESEARCH ESORT SEARCHRES WITHIN
CONTEXT=SEARCH LIST-STATUS BINARY MOVE SPECIAL-USE] Logged in
b select inbox
* FLAGS (\Answered \Flagged \Deleted \Seen \Draft)
* OK [PERMANENTFLAGS (\Answered \Flagged \Deleted \Seen \Draft \*)] Flags
permitted.
* 3 EXISTS
* 0 RECENT
* OK [UNSEEN 1] First unseen.
* OK [UIDVALIDITY 1536038369] UIDs valid
* OK [UIDNEXT 4] Predicted next UID
b OK [READ-WRITE] Select completed (0.000 + 0.000 secs).
b logout
* BYE Logging out
```

```
b OK Logout completed (0.000 + 0.000 secs).
closed
```

On peut maintenant se connecter au serveur IMAP depuis n'importe quel client du réseau local. Pour accéder à votre serveur depuis n'importe où, il faut ouvrir les ports nécessaires dans le pare-feu du routeur.

Configurer l'accès webmail

La plupart du travail est fait, mais on va ajouter un serveur Webmail au serveur de messagerie.

Squirrelmail

Squirrelmail est un Webmail simple et léger pour le serveur et ne demandant pas de bibliothèques de codes additionnelles. **SquirrelMail** est écrit en PHP. Les fonctions de base peuvent être étendues par des plugins. Certains d'entre eux cependant réclament l'application de modifications sur les sources de SquirrelMail pour fonctionner. La configuration se fait simplement à l'aide d'un script perl.

SquirrelMail intègre en standard un carnet d'adresses qui ne nécessite pas de base de données pour fonctionner. Il se contente d'utiliser des fichiers stockés sur le serveur. Il est bien sur possible de configurer l'accès à un annuaire LDAP ou encore d'utiliser une base de données comme **MySql**.

Installer **Apache2** :

```
sudo apt install apache2
```

Ou **nginx**:

```
sudo apt install nginx
```

On peut maintenant installer les packages **squirrelmail**:

En raison de certaines dépendances de PHP5, il faut activer le référentiel **Jessie** avant de procéder à l'installation de **Squirrelmail** :
`sudo vi /etc/apt/sources.list`
ajouter la ligne suivante à la fin :
`deb http://mirrordirector.raspbian.org/raspbian/ jessie main contrib non-free rpi`
Mettre à jour liste des packages: `sudo apt-get update`

```
apt-get install squirrellmail
```

Squirrelmail est livré avec un exemple de fichier de configuration Apache dans `/etc/squirrelmail/apache.conf`. On peut copier ce fichier dans `/etc/apache2/sites-available/squirrelmail` avec la commande :

```
sudo cp /etc/squirrelmail/apache.conf /etc/apache2/sites-
available/squirrelmail.conf
```

puis l'activer en créant un lien dans le répertoire `sites-enabled` avec la commande :

```
sudo ln -s /etc/apache2/sites-available/squirrelmail.conf
/etc/apache2/sites-enabled/squirrelmail.conf
```

Recharger la configuration d'Apache :

```
sudo /etc/init.d/apache2 force-reload
```

Si **squirrelmail** est livré avec une configuration par défaut pour **apache2** ce n'est pas le cas pour **NGINX**. Il faut donc créer un hôte virtuel comme suit :

```
mkdir -p /var/www/www.example.com/web
```

Ensuite, créer une configuration de base de vhost **nginx** pour le vhost www.example.com dans le répertoire `/etc/nginx/sites-available/` comme suit :

```
vi /etc/nginx/sites-available/www.example.com.vhost
```

```
server {
       listen 80;
       server_name www.example.com example.com;
       root /var/www/www.example.com/web;
       if ($http_host != "www.example.com") {
                 rewrite ^ http://www.example.com$request_uri permanent;
       }
       index index.php index.html;
       location = /favicon.ico {
                log_not_found off;
                access_log off;
       }
       location = /robots.txt {
                allow all;
                log_not_found off;
                access_log off;
       }
       # S'assure que les fichiers avec les extensions suivantes ne sont pas
chargés par nginx car nginx afficherait le code source et ces fichiers
peuvent contenir des MOTS DE PASSE !
```

```
        location ~*
\.(engine|inc|info|install|make|module|profile|test|po|sh|.*sql|theme|tpl(\.
php)?|xtmpl)$|^(\..*|Entries.*|Repository|Root|Tag|Template)$|\.php_ {
                deny all;
        }
       #  Refuse toutes les tentatives d'accès aux fichiers cachés tels que
.htaccess, .htpasswd, .DS_Store (Mac).
       location ~ /\. {
                deny all;
                access_log off;
                log_not_found off;
       }
       location ~*  \.(jpg|jpeg|png|gif|css|js|ico)$ {
                expires max;
                log_not_found off;
       }
       location ~ \.php$ {
                try_files $uri =404;
                include /etc/nginx/fastcgi_params;
                fastcgi_pass 127.0.0.1:9000;
                fastcgi_param SCRIPT_FILENAME
$document_root$fastcgi_script_name;
       }
}
```

Pour activer ce vhost, créer un lien symbolique vers celui-ci à partir du répertoire `/etc/nginx/sites-enabled/` :

```
cd /etc/nginx/sites-enabled/
ln -s /etc/nginx/sites-available/www.example.com.vhost www.example.com.vhost
```

Pour utiliser https au lieu de http, il faut ajouter la ligne `fastcgi_param HTTPS ;` à la configuration **SquirrelMail** comme ceci :

```
server {
[...]
        location /squirrelmail {
                root /usr/share/;
                index index.php index.html index.htm;
                location ~ ^/squirrelmail/(.+\.php)$ {
                        try_files $uri =404;
                        root /usr/share/;
                        fastcgi_pass 127.0.0.1:9000;
                        fastcgi_param HTTPS on; # <-- add this line
                        fastcgi_index index.php;
                        fastcgi_param SCRIPT_FILENAME
$document_root$fastcgi_script_name;
                        include /etc/nginx/fastcgi_params;
```

```
                }
                location ~*
^/squirrelmail/(.+\.(jpg|jpeg|gif|css|png|js|ico|html|xml|txt))$ {
                        root /usr/share/;
                }
        }
        location /webmail {
                rewrite ^/* /squirrelmail last;
        }
[...]
}
```

Pour utiliser à la fois http et https, il faut ajouter la section suivante à la section http {} dans `/etc/nginx/nginx.conf` (avant les deux lignes d'inclusion) qui détermine si le visiteur utilise http ou https et définit la variable `$fastcgi_https` en conséquence :

```
vi /etc/nginx/nginx.conf
```

```
[...]
http {
[...]
        ## Detect when HTTPS is used
        map $scheme $fastcgi_https {
          default off;
          https on;
        }
        ##
        # Virtual Host Configs
        ##
        include /etc/nginx/conf.d/*.conf;
        include /etc/nginx/sites-enabled/*;
}
[...]
```

Ensuite, ouvrir le fichier de configuration vhost, et au lieu de `fastcgi_param HTTPS on` ; ajouter la ligne `fastcgi_param HTTPS $fastcgi_https`:

```
vi /etc/nginx/sites-available/www.example.com.vhost
```

```
server {
[...]
        location /squirrelmail {
                root /usr/share/;
                index index.php index.html index.htm;
                location ~ ^/squirrelmail/(.+\.php)$ {
```

```
                        try_files $uri =404;
                        root /usr/share/;
                        fastcgi_pass 127.0.0.1:9000;
                        fastcgi_param HTTPS $fastcgi_https; # <--
add this line
                        fastcgi_index index.php;
                        fastcgi_param SCRIPT_FILENAME
$document_root$fastcgi_script_name;
                        include /etc/nginx/fastcgi_params;
                }
                location ~*
^/squirrelmail/(.+\.(jpg|jpeg|gif|css|png|js|ico|html|xml|txt))$ {
                        root /usr/share/;
                }
        }
        location /webmail {
                rewrite ^/* /squirrelmail last;
        }
[...]
}
```

Recharger **nginx** pour que les modifications prennent effet :

```
/etc/init.d/nginx reload
```

Ensuite, installer SquirrelMail comme suit :

```
apt-get install squirrelmail
```

Il faut maintenant configurer **SquirrelMail** et au moins lui indiquer quel démon POP3-IMAP utiliser:

- choisir `D. Set pre-defined settings for specific IMAP servers`
- choisir `dovecot = Dovecot Secure IMAP server`
- choisir `S Enregistrer les données`
- choisir `Q Quitter`

On peut maintenant trouver **SquirrelMail** dans le répertoire `/usr/share/squirrelmail/`. Il faut maintenant configurer le vhost pour que **nginx** puisse trouver **SquirrelMail** dans ce répertoire.

Ouvrir `/etc/nginx/sites-available/www.example.com.vhost`...

```
vi /etc/nginx/sites-available/www.example.com.vhost
```

... et ajouter la partie suivante au conteneur du serveur {} :

```
server {
```

```
[...]
        location /squirrelmail {
                root /usr/share/;
                index index.php index.html index.htm;
                location ~ ^/squirrelmail/(.+\.php)$ {
                        try_files $uri =404;
                        root /usr/share/;
                        fastcgi_pass 127.0.0.1:9000;
                        fastcgi_index index.php;
                        fastcgi_param SCRIPT_FILENAME
$document_root$fastcgi_script_name;
                        include /etc/nginx/fastcgi_params;
                }
                location ~*
^/squirrelmail/(.+\.(jpg|jpeg|gif|css|png|js|ico|html|xml|txt))$ {
                        root /usr/share/;
                }
        }
        location /webmail {
                rewrite ^/* /squirrelmail last;
        }
[...]
}
```

Recharger nginx :

```
/etc/init.d/nginx reload
```

On peut maintenant accéder à `http://www.example.com/squirrelmail` ou `http://www.example.com/webmail` dans un navigateur, et si tout se passe bien, se connecter à **SquirrelMail** en utilisant l'utilisateur et le mot de passe « pi ».

APC est un cache d'opcode PHP gratuit et ouvert pour la mise en cache et l'optimisation du code intermédiaire PHP. Il est similaire à d'autres cacheurs d'opcodes PHP, tels que **eAccelerator** et **XCache**. Il est fortement recommandé d'installer un de ceux-ci pour accélérer le chargement des pages PHP:

```
apt-get install php-apc
```

Lorsqu'on utilise **PHP-FPM** comme démon **FastCGI** (comme dans Installation de Nginx avec PHP5 (et PHP-FPM) et le support MySQL sur Ubuntu 11.04), il faut le redémarrer ainsi:
`\\/etc/init.d/php5-fpm restart`

Si le programme **spawn-fcgi** de **lighttpd** est utilisé comme démon FastCGI il faut tuer le processus spawn-fcgi actuel (exécuté sur le port 9000) et en créer un nouveau.

* taper `netstat -tap` pour récupérer le PID du processus spawn-fcgi actuel
* arrêter le processus en cours: `kill -9 <PID du processus>`
* créer un nouveau processus spawn-fcgi: `/usr/bin/spawn-fcgi -a 127.0.0.1 -p 9000 -u www-data -g www-data -f /usr/bin/php5-cgi -P /var/run/fastcgi-php.pid`

Roundcube

Roundcube est un logiciel de webmail moderne, gratuit et open-source.

Le grand avantage de **Roundcube** par rapport aux autres webmails est qu'il est disponible directement dans les dépôts de Debian, et donc de Raspberry Pi OS.

Il faut d'abord installer un serveur MySQL (MariaDB) pour stocker la base de données **Roundcube**:

```
sudo apt install mariadb-server
```

Ensuite suivre les étapes suivantes pour définir un mot de passe root et créer un utilisateur **Roundcube**:

Se connecter à la console MySQL en root:

```
sudo mysql -uroot
```

Définir le mot de passe de la racine :

```
use mysql ; UPDATE user SET password=PASSWORD('MotDePasse'), plugin='' WHERE User='root' AND Host = 'localhost' ; FLUSH PRIVILEGES ;
```

Remplacer « MotDePasse » par un mot de passe sécurisé.

Créer un nouvel utilisateur pour **Roundcube**:

```
CREATE USER 'roundcube'@'localhost' IDENTIFIED BY 'MotDePasse' ;
```

Remplacer « MotDePasse » par le mot de passe de votre choix.

Créer la base de données **Roundcube**:

```
CREATE DATABASE roundcubemail;
```

Donner tous les privilèges à l'utilisateur Roundcube sur la base de données **Roundcube**:

```
GRANT ALL PRIVILEGES ON roundcubemail.* to 'roundcube'@'localhost' ;
```

Quitter la console **MySQL** :

```
FLUSH PRIVILEGES ;
quit
```

Le serveur de base de données est prêt.

Pour installer **roudcube**, entrez la commande suivante :

```
sudo apt install roundcube roundcube-plugins
```

Cela installera aussi automatiquement toutes les autres dépendances (principalement **Apache**, **PHP** et le client **MySQL**).

L'assistant d'installation posera ces questions sur le serveur **MySQL** :

- **Configurer la base de données avec dbconfig-common**: oui.
- **Mot de passe de l'application MySQL pour Roundcube**: <mot de passe d'utilisateur Roundcube>.
- **Mot de passe administrateur de la base de données**: <mot de passe root MySQL>

Maintenant, éditer la configuration **Apache** pour **Roundcube** afin d'activer l'application web :

```
sudo vi /etc/apache2/conf-enabled/roundcube.conf
```

Décommenter la première ligne :

```
Alias /roundcube /var/lib/roundcube
```

Puis redémarrer le serveur web :

```
sudo service apache2 restart
```

Pour vérifier aller à http://[IP-DU-RASPBERRYPI]/roundcube pour voir l'interface web.

Si il y a une erreur, on peut relancer l'assistant d'installation avec cette commande :

```
sudo dpkg-reconfigure roundcube-core
```

On peut maintenant se connecter avec les informations d'identification créées à l'étape précédente, ou avec le compte « pi ».

Fichiers de connexion et de configuration

Si il y a des erreurs, ou pour aller plus loin, voici le résumé de l'emplacement des fichiers.

Application	Commentaire	Emplacements
Postfix	Pour envoyer et recevoir des e-mails, c'est le cœur du serveur de messagerie.	Configuration principale: `/etc/postfix/main.cf` Configuration des processus de Postfix : `/etc/postfix/master.cf` Logs des emails, et des erreurs : `/var/log/mail.log`
Dovecot	Pour gérer les connexions IMAP avec une couche de sécurité SASL.	Configuration principale: `/etc/dovecot/dovecot.conf` Sous-dossier contient plusieurs fichiers avec chaque partie de la configuration: `/etc/dovecot/conf.d/` Fichiers de logs (Dovecot n'a pas de fichier de log spécifique, il utilise le fichier syslog principal): `/var/log/syslog`
Apache	Pour faire tourner le webmail	Configuration principale: `/etc/apache2/apache2.conf` Configuration des services (comme Roundcube.conf): `/etc/apache2/conf-enabled/` Configuration pour tout site web: `/etc/apache2/sites-enabled/` Fichiers de logs: `/var/log/apache/error.log`
Rondcube	webmail	Configuration principale: `/etc/roundcube/config.inc.php` Fichiers de logs: `/var/log/roundcube/errors`