# Création d'un serveur de messagerie dans un conteneur

**Table of Contents**



Cet atelier a pour objet de tester la création d'un serveur autohébergé dans un conteneur **LXC**

### Création du conteneur

Sur le Host créer un nouveau conteneur dans lequel exécuter le serveur de messagerie :

Le script de saltstack **salt_tarball** permet de créer un conteneur à partir d'une archive tar. Le site https://jenkins.linuxcontainers.org/view/Images/ propose des images pour LXC et LXD construites avec **distrobuilder**.

**distrobuilder** est utilisé pour construire toutes les images officielles disponibles.

La définition de l'image est un document YAML qui décrit la source de l'image, son gestionnaire de packages, les packages à installer/supprimer pour des variantes d'image spécifiques, des versions et des architectures du système d'exploitation, ainsi que des fichiers supplémentaires à générer et des actions arbitraires à exécuter dans le cadre de le processus de création d'images.

La sortie est soit un système de fichiers racine simple, une image LXD ou une image LXC.

La version actuelle peut également être installée directement avec :

```
go get -v -x github.com/lxc/distrobuilder/distrobuilder
```

```
apt-get -y install lxc
wget
https://jenkins.linuxcontainers.org/view/Images/job/image-debian/architectur
e=arm64,release=bullseye,variant=default/lastSuccessfulBuild/artifact/rootfs
.tar.xz
https://raw.githubusercontent.com/saltstack/salt/develop/salt/templates/lxc/
salt_tarball
```

```
chmod 777 salt_tarball
lxc-update-config -c salt_tarball
lxc-create -n smail -t ~/salt_tarball -- --network_link lxcbr0 --imgtar
./rootfs.tar.xz
```

Lui donner une adresse IP statique via **dnsmasq** :

```
echo "dhcp-host=smail,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
```

```
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 Postfix pour envoyer des e-mails

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.

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).

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`

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.

### 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.

### 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]
```

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.

### Installer le webmail Squirrelmail

Installer les packages **nginx**:

```
sudo apt-get install nginx
```

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/*;
}
[...]
```

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

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
````

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

/etc/init.d/nginx reload

```
Ensuite, installer SquirrelMail comme suit :

<WRAP important>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`</WRAP>
```

apt-get install squirrellmail

```
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/`.

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
```