

Postfix+Courier-IMAP+MySQL for multiple domains HOWTO

AUTHORS: Kirby Menzel and Lucas Peet

Acknowledgements: This howto relies heavily on the howto by Keith Matthews located at http://www.sweeney.demon.co.uk/pfix_imap_virtual.html as well as the one at http://kummefryser.dk/HOWTO/mail/postfix_mysql.html.

Both are good reads, but most relevant information is now included here. After all, part of our goal was to make it so you didn't have to search all over the internet like we did. Also, thanks to Mark Garfias for catching some errors quickly before they became major problems for anyone else using the howto. :)

Purpose:

The purpose behind this document is to create a mail server solution that includes postfix as the MTA, Courier-IMAP as the IMAP/POP server and MySQL for all potentially dynamic configuration. The goal is to have completely virtual users and domains.

`jimbob@domain1.com != jimbob@domain2.com`. This means creating a separate namespace for each domain. Potential advantages of doing all of this are pretty clear and left up to the reader to determine.

Related Docs:

MySQL: <http://mysql.com/documentation/index.html>

Postfix: <http://www.postfix.org/docs.html>

Courier-IMAP: <http://www.inter7.com/courierimap/>

The Theory:

Now, the theory behind what we are doing here is pretty simple.

The first thought is how to create that separate namespace. Using Postfix virtual delivery agent, the problem is relatively trivialized. You just deliver mail to the appropriate maildir for the appropriate address. Postfix really doesn't care that you have multiple people named jimbob because it looks at the entire address. Separate namespace for Courier gets a little trickier. You could ask all of your users to login as `user@domain.com` or `user.domain.com` or even `user.somenumber` and this is the best solution we've come up with so far. The problem with `user@domain.com` logins is that drain damaged MUA's may interpret that to mean one's complete address is `"user@domain.com@domain.com"`. Needless to say, this is bad. Of course, the same faulty MUA's might cause problems with thinking `"user.domain.com@domain.com"` is the address... I'm really not sure on either of these because we haven't run across anything quite so broken in our testing.

The Solution:

INSTALLATION:

We assume that, for the most part, you know how to handle the basics of installing MySQL, Postfix, and Courier-IMAP. If you do not, please consult the documentation that comes with them. It's really quite good. Remember that in both Postfix and Courier-IMAP, you must enable MySQL support. This means that you must first install MySQL, then the others. If you use an rpm-based Linux such as RedHat or Mandrake, make sure to build from an srpm so that you can enable what you need. If you use an xBSD, the Postfix port has a nice little text-based gui for you to select what you want compiled in, but you must first use `make, then make install`. Under xBSD for Courier-IMAP, read the Makefile in the port to learn how to enable MySQL support.

Instructions under RedHat

Note this must be done as a non-root user!

Install the source:

```
rpm -Uvh postfix-...src.rpm
```

Depending on the version of MySQL you have installed (yes, they're different!) there's a different export line you need to do. This is because the 'real' MySQL RPM (from mysql.com) installs the files in a different location than the one installed by RedHat. Why, I have NO CLUE! (KM: Becuase RedHat and MySQL both believe that their way is the only right way?) Anyway, here it is:

If you have the 'real' MySQL:

```
export POSTFIX_MYSQL=1
```

If you have the RedHat MySQL:

```
export POSTFIX_REDHAT_MYSQL=1
```

If you're not sure which version you have, do:

```
rpm -q mysql
```

If it showed you the package, you have the RedHat version.

If you got 'package not installed', then try:

```
rpm -q MySQL
```

If it showed you the package, you have the MySQL version.

If you got 'package not installed', I suggest you install it! :)

The Database:

Here is the quick description of the database:

```
mysql> show tables;
```

```
+-----+
| Tables_in_maildb |
+-----+
| transport        |
| users            |
| virtual          |
+-----+
```

```
3 rows in set (0.00 sec)
```

```
mysql> describe transport;
```

```
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| domain     | varchar(128)  |      | PRI |          |       |
| transport  | varchar(128)  |      | MUL |          |       |
+-----+-----+-----+-----+-----+-----+
```

```
2 rows in set (0.00 sec)
```

```
mysql> describe virtual;
```

```
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| address    | varchar(255)  |      | PRI |          |       |
| goto       | varchar(255)  |      |     |          |       |
+-----+-----+-----+-----+-----+-----+
```

```
2 rows in set (0.01 sec)
```

```
mysql> describe users;
```

```
+-----+-----+-----+-----+-----+-----+
```

Description of fields in transport:

domainAny domain you host. Including those you want to deliver to using local: as the transport.
transportThe transport method. All transport methods are legal, but usually either "local:" for local transport or "virtual:" for the virtual transport agent.

Description of fields in virtual:

addressThis is the virtual address `address@domain.tld` which will be forwarded to the address in `goto`. For Postfix-style virtuals, this will also be the domain name.
gotoWhere the virtual address above goes. This can be a comma-delimited list of addresses. It can include simply "user" entries for local users (root, postmaster, etc.), or, more commonly, complete `user@domain.tld` entries for users in virtual domains on your machine or for going out to other domains.

Description of fields in users:

idusername. Either `user.domain.tld` or `user@domain.tld`
addressemail address.
`user@domain.com`

(if you use the second form above, this column can be omitted.)
cryptThe `crypt()` form of the password. The easiest way to accomplish this is to use `encrypt('password')` in your insert query.
clearClear text password. (for support purposes or Cram-MD5) You really only need one of the two password forms. If you choose that one to be clear, make sure ALL of your users can handle Cram-MD5 authentication.
nameUsers real name or whatever string you feel belongs here. (not required unless you tell Courier it's there.)
uidvirtual uid
gidvirtual gid (hint to make your life simpler: Make each domain have one and only one gid)
homeYou can set this to just about anything above your `maildir` directory. Easiest (but not very secure) is to set it to `"/`. If you are running postfix `chroot`, this needs to be somewhere inside of the jail. For instance, `"/var/spool/postfix/`. And, yes, the `maildirs` must all be subdirectories of this directory.
domainuser's domain name. This is far from necessary but comes in handy for support/programming purposes.
maildirThis is the users `maildir`. Use the full path. You don't have to, but use the full path. If this is a `maildir`, make sure that you include the trailing slash. (e.g. `.../Maildir/` and NOT `.../Maildir`) Of course, if you want to use `mbox` (why, oh why?!) then you would need to exclude the slash.
imapokThis field would allow one to prevent users from accessing their mail by setting it to 0 if you use the appropriate Courier setting in `authmysqlrc` of `"MYSQL_WHERE_CLAUSE=imapok"` This could be useful for torturing your users, among other things.
bool1Same as above.
bool2ditto. (yes, Kirby is an aspiring BOFH, how'd you guess?)

Transport table:

The transport table is your transport map. It tells postfix what transport agent to use for each domain. It is not absolutely necessary if you want all of your users to be in the database. We do, however, highly recommend using a transport map. Having at least one domain handled with the local delivery agent means, among other things, simplicity if you have to deal with mailing lists later and that any users added to the system as system users will automatically receive mail properly. Even if you have domains explicitly defined in the `mydestinations` config option for postfix, you should make sure that every domain has a corresponding entry in the transport map. Another nice feature of the transport map is that you do not have to define `mydestinations` explicitly, simply add `$transport_maps` to `mydestinations` and you have all your domains added cleanly and effectively without having to reload Postfix when you add a domain.

Virtual table:

The virtual table is your virtual map. It is similar to aliases, but different. An entry in virtual cannot point to an executable or a file. It is simply address to address mapping. However, this doesn't mean it has to be one to one. You can have a single virtual alias point to multiple addresses in a comma-delimited list. The important thing to remember is that you can have either sendmail-style or postfix-style virtual domains. Under sendmail-style, all local aliases and users exist in the virtual domain's namespace as well. In other words, a local user jimbo would be able to receive jimbo@virtual1.com, jimbo@virtual2.com, and jimbo@localmachinedomain.com all at the same time. This can cause a problem when hosting many domains (or only a few if you have more than one user named JimBob.) Enter postfix-style virtuals. With a postfix-style virtual, the virtual domain doesn't share anything. You must define every alias and/or user within that domain. One caveat is that this means root and postmaster aliases must be defined for every postfix-style virtual. In order to implement a postfix-style virtual, you will need one line in your virtual map that has the domain name on the left, and some random text on the right. One handy way of doing this in a database virtual table is to have an entry where address is the domain name and goto is the name of the domain owner, or some other informative little piece of text. A nice convention is to use sendmail-style virtuals on all of your domains that are delivered with the local delivery agent and postfix-style on all of your domains delivered with the virtual agent. For a more thorough description of the virtual table and sendmail-style vs. postfix-style virtuals, please read "man 5 virtual".

Users table:

The users table is the heart and soul of this whole thing. Transport and Virtual can easily be done as text files, but the users table makes this all worthwhile. While the other two tables are used only by Postfix, this table is shared between Postfix and Courier. That leads to interesting decisions in design. First of all, some will note the different id and address fields. In some cases, these fields may be identical. In that case, remove the address field, and replace address in all postfix config files with id. The reason there are two different fields is so you can use user.domain.tld, or user.somenumber or something along those lines rather than user@domain.tld as a username. If you plan on using user@domain.tld for logins and never changing it, don't waste space on an address field. Another important thing to remember is that the encrypted form of the password must be encrypted using the OS crypt(), not with MySQL's default encryption algorithm. This can be done by using crypt=encrypt('password') in your insert or update statement. You can increase the security of this by giving encrypt a second argument of a two-character salt. If you write a user management program to handle your data additions and updates, then creating it randomly will be useful. You can then use encrypt('password','salt') for the improved security. You also should note that you can have as many or as few (including none) of the fields beginning with "imapok" above. Those fields are not required, but can come in handy as mentioned in the table description. Also, remember that if you would like to use the maildir quota extension in courier-imap, you will need to add a quota field to hold that. Probably an unsigned int would be more than sufficient.

Easy Database creation:

For an easy way to create the database, copy the following into a file and at the command line do "mysql -u root -p filename".

```
#First Create the Database
CREATE DATABASE maildb;
use maildb;
#
# Table structure for table 'transport'
#
CREATE TABLE transport (
  domain varchar(128) NOT NULL default ' ',
  transport varchar(128) NOT NULL default ' ',
  UNIQUE KEY domain (domain)
) TYPE=MyISAM;
#
# Table structure for table 'users'
#
CREATE TABLE users (
  id varchar(128) NOT NULL default ' ',
  address varchar(128) NOT NULL default ' ',
  crypt varchar(128) NOT NULL default ' ',
  clear varchar(128) NOT NULL default ' ',
  name varchar(128) NOT NULL default ' ',
  uid smallint(5) unsigned NOT NULL default '1000',
  gid smallint(5) unsigned NOT NULL default '1000',
  home varchar(128) NOT NULL default '/',
  domain varchar(128) NOT NULL default ' ',
  maildir varchar(255) NOT NULL default ' ',
  imapok tinyint(3) unsigned NOT NULL default '1',
  bool1 tinyint(3) unsigned NOT NULL default '1',
  bool2 tinyint(3) unsigned NOT NULL default '1',
  PRIMARY KEY (id),
  UNIQUE KEY id (id),
  UNIQUE KEY address (address),
  KEY id_2 (id),
  KEY address_2 (address)
) TYPE=MyISAM;
#
# Table structure for table 'virtual'
#
CREATE TABLE virtual (
  address varchar(255) NOT NULL default ' ',
  goto varchar(255) NOT NULL default ' ',
  UNIQUE KEY address (address)
) TYPE=MyISAM;
```

Further Notes on the Database:

You might, after looking at the above, be wondering why on a few data type and indexing choices. Well, according to the mysql docs, varchar is faster in lookups and takes less space than char, so that's why it is used for all strings here. As for the size of integers, it was just logical choosing based on potential size of the values. In the case of the essentially boolean values, even the tinyint given to them is too much. These could easily be defined as "int(1) unsigned" if MySQL allows. You could also go with ENUM as a data type for these. See MySQL docs for more details. As for indexing and such, it's a good idea to have both id and address unique and indexed for faster lookups since almost every select performed on the users table will include either id or address in the where clause. On transport, it's a good idea to index the domain field. On virtual, you definitely want address indexed. Your virtual table will get very large, very quickly, when you consider that every virtual domain will have three minimum entries. As you can see, the general idea is to index any field that is likely to show up in a where clause in database query. One other thing on the database: If you run Postfix in a chroot jail, and you are using localhost as your mysql server, you might want to add a hardlink to the mysql socket somewhere inside the jail/sandbox you are running in. Of course, this will likely break when you restart the MySQL server. Instead, it would be better

In Postfix:

If you don't want all users, including those locally defined to be in the database, you'll want to set up a transport map. We used a database-driven

transport map as described above. We will assume transport to be either "virtual:" for virtual delivery or "local:" for local delivery. Other methods are available, but are beyond the scope of this howto. The files listed here will be in your postfix config directory. On FreeBSD, that should be `/usr/local/etc/postfix`. On OpenBSD and most Linux distros

`/etc/postfix`.

```
# in main.cf
transport_maps=mysql:/usr/local/etc/postfix/transport.cf
virtual_mailbox_maps=mysql:/usr/local/etc/postfix/mysql_virt.cf
virtual_uid_maps=mysql:/usr/local/etc/postfix/uids.cf
virtual_gid_maps=mysql:/usr/local/etc/postfix/gids.cf
virtual_mailbox_base=/
#You might want to change this to something more secure. In untested theory, you
mydestination = $mydomain, $myhostname, $transport_maps
virtual_maps =mysql:/usr/local/etc/postfix/virtual.cf
```

```
# in master.cf you need to add this line if it isn't already there.
virtual    unix    -        n        n        -        -        virtual
```

```
# transport.cf
user=postfix
password=whatever
dbname=maildb
table=transport
select_field=transport
where_field=domain
hosts=localhost
```

```
# mysql_virt.cf
user=postfix
password= whatever
dbname=maildb
table=users
select_field=maildir
where_field=address
hosts=localhost
```

```
# uids.cf
user=postfix
password=whatever
dbname=maildb
table=users
select_field=uid
where_field=address
hosts=localhost
```

```
# gids.cf

user=postfix
password=whatever
dbname=maildb
table=users
select_field=gid
where_field=address
hosts=localhost
```

```
# virtual.cf

user=postfix
password=whatever
dbname=maildb
table=virtual
select_field=goto
where_field=address
hosts=localhost
```

In Courier-IMAP:

The file to edit in Courier-IMAP is "authmysqlrc". You'll want to define the following within it appropriately. I've noticed that the sample config file is well-commented on my installation. Hopefully the same is true for you. If you do not see a authmysqlrc.dist or some sample authmysqlrc, the below should give you a working configuration. There are a few other options, but we will not cover them here as they are unnecessary for our purposes. One of note is MYSQL_QUOTA_FIELD which allows you to use the courier quota extensions to maildir. The files below should be located in your courier config directory. Under FreeBSD that is /usr/local/etc/courier-imap. Under OpenBSD and most Linux distros /etc/courier-imap.

```
#Contents of authmysqlrc
MYSQL_SERVER          ; localhost
#your mysql server
MYSQL_USERNAME        courier
MYSQL_PASSWORD        whatever
MYSQL_SOCKET          /tmp/mysql.sock #necessary if you are on localhost
MYSQL_DATABASE        maildb
MYSQL_USER_TABLE      users
MYSQL_CRYPT_PWFIELD   crypt
MYSQL_CLEAR_PWFIELD   clear
MYSQL_UID_FIELD       uid
MYSQL_GID_FIELD       gid
MYSQL_LOGIN_FIELD     id
MYSQL_HOME_FIELD     home
MYSQL_NAME_FIELD      name
MYSQL_MAILDIR_FIELD   maildir
MYSQL_WHERE_CLAUSE    imapok=1 AND bool1=1 AND bool2=1
```

```
#Relevant settings in authdaemonrc
authmodulelist="authmysql authpam" #replace authpam with whatever your local auth
is, authpwd, authsas1, whatever.
```

If you only want users who are in the database to login, then only use authmysql above
version="authdaemon.mysql"

In the File System:

Virtual user needs to have read and execute permissions on all parent directories up to /. (or /var/spool/postfix/ if running in chroot). Virtual user needs to have write permissions on the mailbox directory (or file if using mbox format). Permissions on the user's mailbox files should be 770, and chown uid.gid.

Sample file system under chrooted version:

```
/var/spool/postfix/virtual:  
drwxr-xr-x    4 postfix postfix    4096 Apr  7 11:25 virtual  
/var/spool/postfix/virtual/lucastek:  
drwxrwxr-x    4 postfix postfix    4096 Apr  6 20:04 lucastek.com  
/var/spool/postfix/virtual/lucastek/users  
drwxrwx---   12 5001    5001    4096 Apr  6 19:53 user1  
drwxrwx---    6 5002    5002    4096 Apr  7 11:22 user2
```

You might also want to create the maildir rather than waiting on Postfix to create upon first delivery. This can be done using the handy utility maildirmake which comes with Courier-IMAP.

Example usage assuming the above filesystem and that Maildir is the name of the maildir under the user1 directory and has not yet been created:

```
maildirmake /var/spool/postfix/virtual/lucastek/user1/Maildir
```

Yes, it really is that simple. Of course, after doing this don't forget to use chown -R to change the ownership on everything in user1.

Conclusion:

Hopefully, this howto was useful to you. If you feel there was a lack or would like to see something added, please email the authors and we will gladly consider any additions contributed. Also keep in mind that the man pages are your friends. Before asking a question about a specific utility, check the man page. You just might find the solution. Of course, if you think that solution belongs here, just add it :)