PostfixAdmin-on Ubuntu 20.04 Mail Server

PostfixAdmin Features

- manage mailboxes, virtual domains, and aliases
- vacation/out-of-office messages (Personally I think it's better done in Roundcube webmail)
- alias domains (forwarding one domain to another with recipient validation)
- users can manage their own mailbox (change alias, password and vacation message)
- quota support for single mailboxes and total quota of a domain
- fetchmail integration: You can fetch emails from your original email address to your new email address.
- command-line client postfixadmin-cli for those who don't want to click around in a web interface 😉

Note: Once you finish part 3, you can no longer use local Unix accounts as email addresses. You must create email addresses from the PostfixAdmin web interface.

Prerequisites

It's required that you have followed part 1 and part 2 of this tutorial series before continuing to read this article. If you followed mail server tutorials on other websites, I recommend purging your configurations (`sudo apt purge postfix dovecot-core`) and start over with my tutorial series, so you are not going to be confused by different setup processes.

Once the above requirements are met, let's install and configure PostfixAdmin.

Step 1: Install MariaDB Database Server

PostfixAdmin is written in PHP and requires a database (MySQL/MariaDB, PostgreSQL or SQLite). This article will use MariaDB database, which is a drop-in replacement for MySQL. It is developed by former members of MySQL team who are concerned that Oracle might turn MySQL into a closed-source product. Enter the following command to install MariaDB on Ubuntu 20.04.

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

After it's installed, MariaDB server should be automatically started. Use **systemctl** to check its status.

```
systemctl status mariadb
```

Output:

```
● mariadb.service - MariaDB 10.3.22 database server

     Loaded: loaded (/lib/systemd/system/mariadb.service; enabled; vendor preset: enabled)

     Active: active (running) since Fri 2020-04-10 14:19:16 UTC; 18s ago

       Docs: man:mysqld(8)

             https://mariadb.com/kb/en/library/systemd/

   Main PID: 9161 (mysqld)

     Status: "Taking your SQL requests now..."

      Tasks: 31 (limit: 9451)

     Memory: 64.7M

     CGroup: /system.slice/mariadb.service

             └─9161 /usr/sbin/mysqld
```

If it’s not running, start it with this command:

```
sudo systemctl start mariadb
```

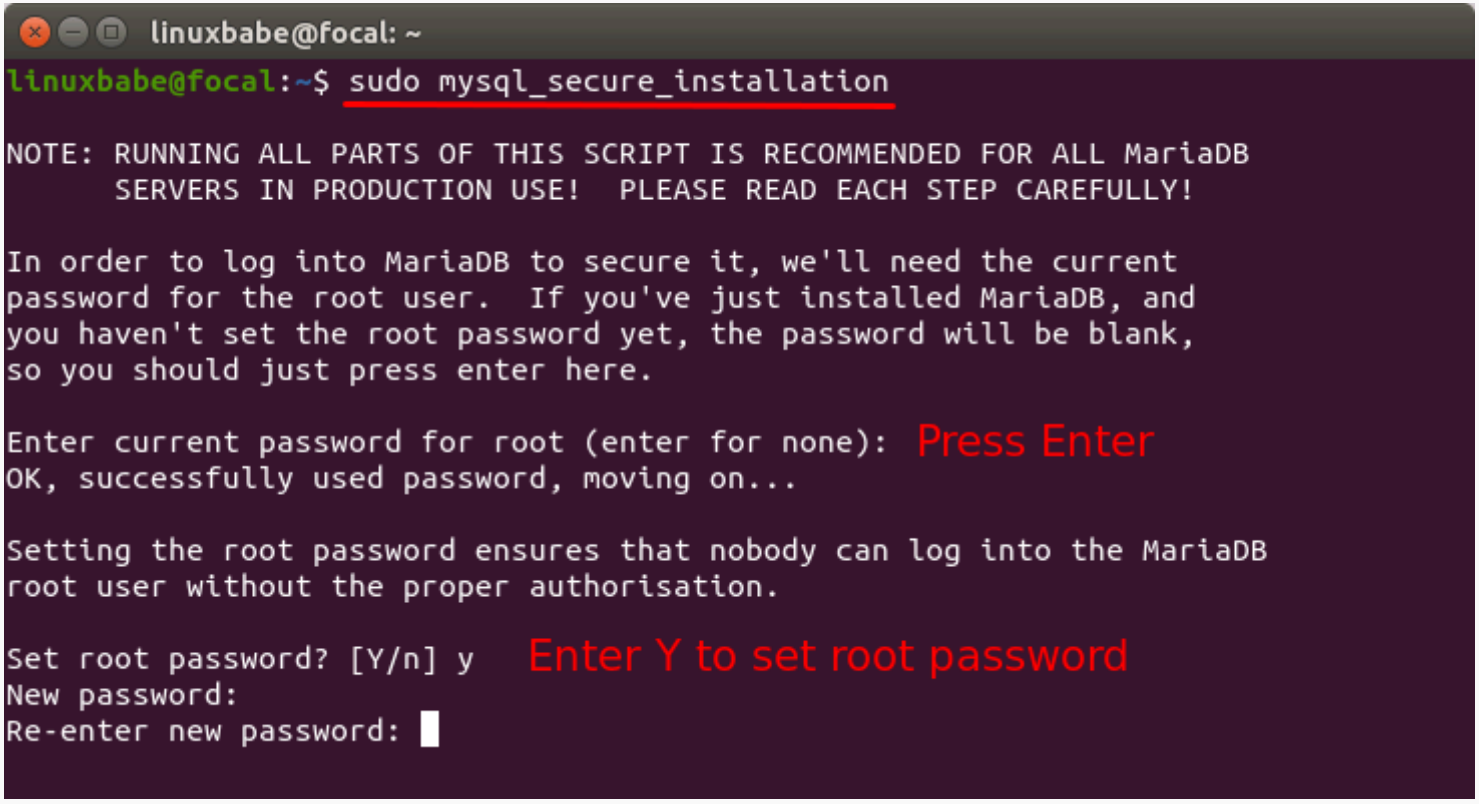
To enable MariaDB to automatically start at boot time, run

```
sudo systemctl enable mariadb
```

Now run the post-installation security script.

```
sudo mysql_secure_installation
```

When it asks you to enter MariaDB root password, press Enter key as the root password isn't set yet. Then enter **y** to set the root password for MariaDB server.



Next, you can press Enter to answer all remaining questions, which will remove anonymous user, disable remote root login and remove test database. This step is a basic requirement for MariaDB database security. (Notice that Y is capitalized, which means it is the default answer.)

```
By default, a MariaDB installation has an anonymous user, allowing anyone
to log into MariaDB without having to have a user account created for
them.  This is intended only for testing, and to make the installation
go a bit smoother.  You should remove them before moving into a
production environment.

Remove anonymous users? [Y/n]  Press Enter
 ... Success!

Normally, root should only be allowed to connect from 'localhost'.  This
ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? [Y/n]  Press Enter
 ... Success!

By default, MariaDB comes with a database named 'test' that anyone can
access.  This is also intended only for testing, and should be removed
before moving into a production environment.

Remove test database and access to it? [Y/n]  Press Enter
 - Dropping test database...
 ... Success!
 - Removing privileges on test database...
 ... Success!

Reloading the privilege tables will ensure that all changes made so far
will take effect immediately.

Reload privilege tables now? [Y/n]  Press Enter
 ... Success!

Cleaning up...

All done!  If you've completed all of the above steps, your MariaDB
installation should now be secure.

Thanks for using MariaDB!
```

Step 2: Install PostfixAdmin on Ubuntu 20.04 Server

Log into your mail server. Because some readers use MariaDB server, while others use MySQL, which makes things complicated, so before installing PostfixAdmin, we install the `dbconfig-no-thanks` package to prevent the `postfixadmin` package from launching the database configure wizard.

```
sudo apt install dbconfig-no-thanks
```

Then install PostfixAdmin from the default Ubuntu software repository.

```
sudo apt install postfixadmin
```

Note: If you have previously installed `mysql-server` on Ubuntu, the installation of PostfixAdmin will probably remove the `mysql-server` package from your system. You can re-install it by running the following command.

```
sudo apt install mysql-server
```

Now we need to remove the `dbconfig-no-thanks` package.

```
sudo apt remove dbconfig-no-thanks
```

Then launch the database configure wizard for PostfixAdmin.

```
sudo dpkg-reconfigure postfixadmin
```

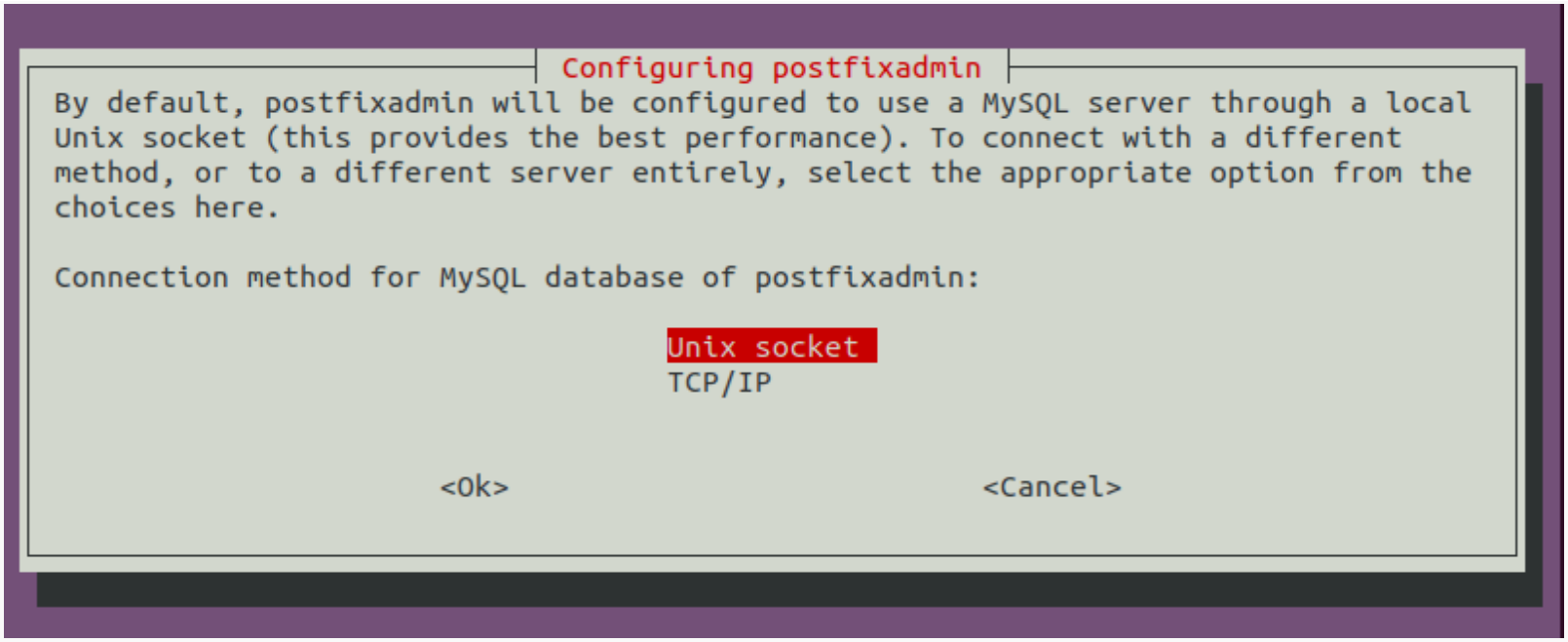
During the installation, you will be asked if you want to reinstall database for PostfixAdmin. This simply means creating a database named `postfixadmin`, it won't remove your existing databases. Press the Tab key to choose **Yes**.

```
Configuring postfixadmin

Since you are reconfiguring postfixadmin, you may also want to reinstall the
database which it uses.

If you wish to reinstall the database for postfixadmin, you should select this
option. If you do not wish to do so (if you are reconfiguring the package for
unrelated reasons), you should not select this option.

Warning: if you opt to reinstall the database and install it under a name that
already exists, the old database will be dropped without further questions. In
that case a backup of the original database is made in /var/tmp/.

Warning: if you change the name of the database, the old database will not be
removed. If you change the name of the user that connects to the database, the
privileges of the original user will not be revoked.

Reinstall database for postfixadmin?

                    <Yes>                    <No>
```

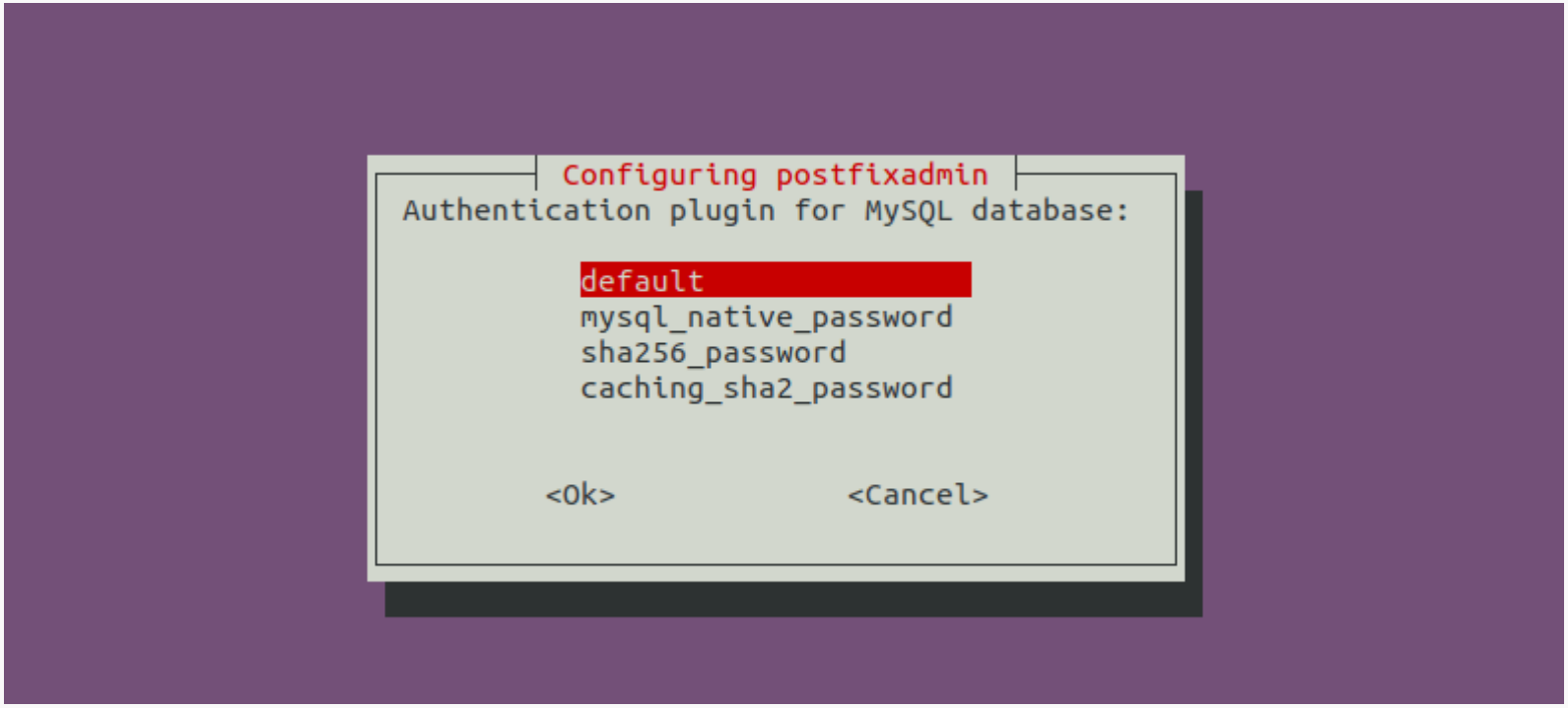
Then select the default database type: `mysql`, if you use MySQL or MariaDB.

```
Configuring postfixadmin
The postfixadmin package can be configured to use one of several database
types. Below, you will be presented with the available choices.

If other database types are supported by postfixadmin but not shown here, the
reason for their omission is that the corresponding dbconfig-<database type>
packages are not installed. If you know that you want the package to use
another supported database type, your best option is to back out of the
dbconfig-common questions and opt out of dbconfig-common assistance for this
package for now. Install your preferred dbconfig-<database type> option from
the list in the package dependencies, and then "dpkg-reconfigure postfixadmin"
to select it.

Database type to be used by postfixadmin:

                                mysql
                                pgsql


                  <Ok>                        <Cancel>
```

Next, choose the default connection method: `Unix socket`.



Then choose the default authentication plugin for MySQL/MariaDB.



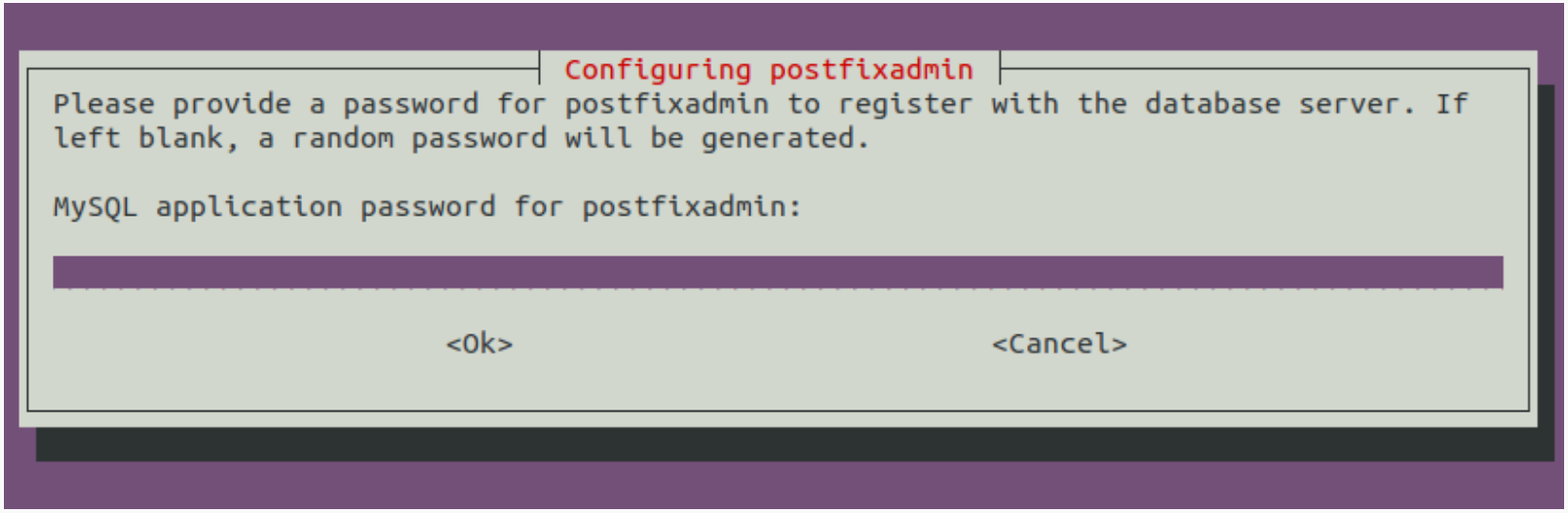
Press Enter to choose the default database name for PostfixAdmin.

```
Configuring postfixadmin
Please provide a name for the MySQL database to be used by postfixadmin.

MySQL database name for postfixadmin:

postfixadmin

<Ok>                    <Cancel>
```

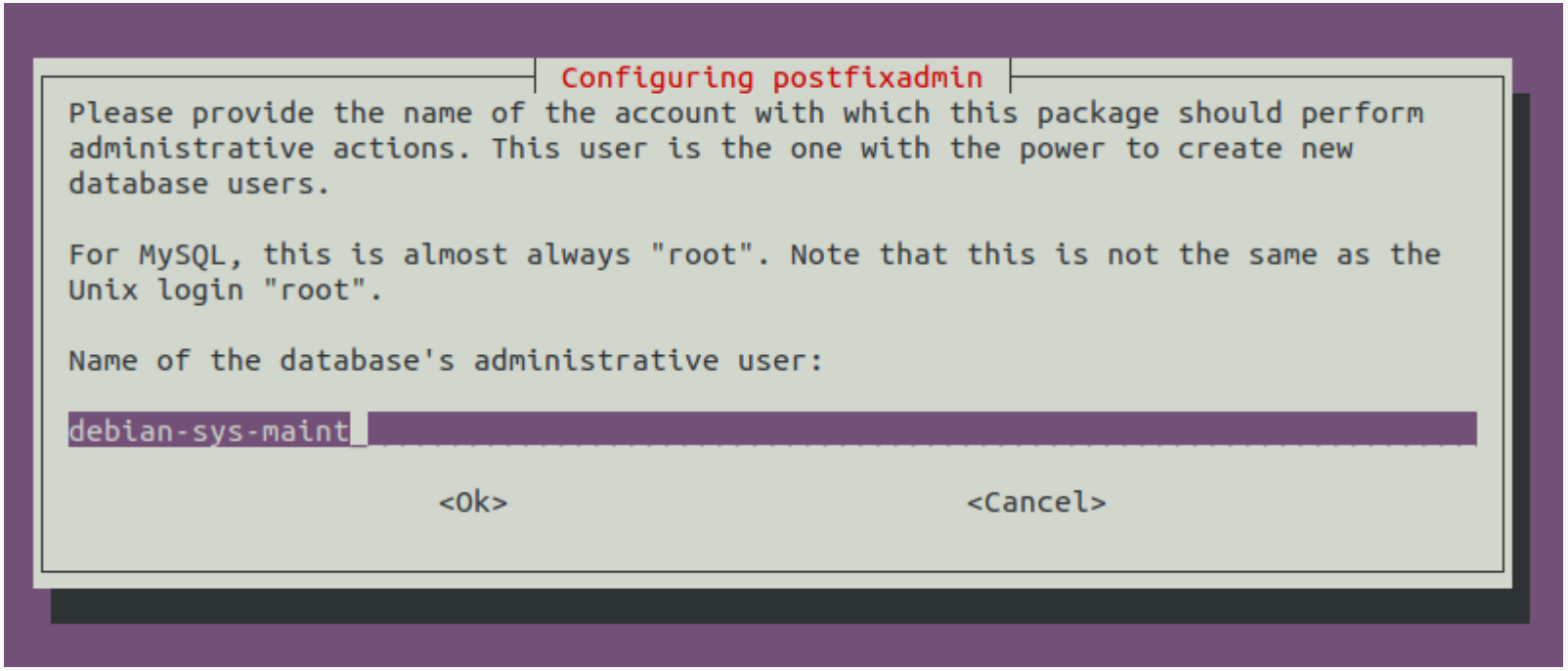
Press Enter to choose the default database username for PostfixAdmin.

```
Configuring postfixadmin
Please provide a MySQL username for postfixadmin to register with the database
server. A MySQL user is not necessarily the same as a system login, especially if
the database is on a remote server.

This is the user which will own the database, tables, and other objects to be
created by this installation. This user will have complete freedom to insert,
change, or delete data in the database.

If your username contains an @, you need to specify the domain as well (see
below).

Advanced usage: if you need to define the domain that the user will log in from,
you can write "username@domain".

MySQL username for postfixadmin:

postfixadmin@localhost

<Ok>                    <Cancel>
```

After that, you need to set a password for this user. Note that the password should not contain the # character, or you might not be able to log in later.



Finally, choose the default database administrative user.



After PostfixAdmin is installed, you can log in to MySQL/MariaDB console with the following command. You will need to enter the password for the `postfixadmin` user.

```
mysql -u postfixadmin -p
```

And you can check what databases the user has permissions to access with the following command.

```
SHOW DATABASES;
```

Output:

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

| Database           |

+--------------------+

| information_schema |

| postfixadmin       |

+--------------------+

2 rows in set (0.002 sec)
```

By default, the `postfixadmin` database contains no tables. You can log out of the MySQL/MariaDB console with the following command.

```
EXIT;
```

The installation will also create two configuration files: `/etc/dbconfig-common/postfixadmin.conf` and `/etc/postfixadmin/dbconfig.inc.php`, both of which contain the database access settings, including the database username and password. We need to change the database type from `mysql` to `mysqli` in both of the two files.

```
sudo nano /etc/dbconfig-common/postfixadmin.conf
```

Change

```
dbc_dbtype='mysql'
```

to

```
dbc_dbtype='mysqli'
```

Then edit the second file.

```
sudo nano /etc/postfixadmin/dbconfig.inc.php
```

Change

```
$dbtype='mysql';
```

to

```
$dbtype='mysqli';
```

The web files are installed under `/usr/share/postfixadmin/` directory, which is owned by root. PostfixAdmin requires a `templates_c` directory, so create it.

```
sudo mkdir /usr/share/postfixadmin/templates_c
```

We need to give `www-data` user read, write and execute permissions on this directory with the following command.

```
sudo setfacl -R -m u:www-data:rwx /usr/share/postfixadmin/templates_c/
```

If your system can't find the `setfacl` command, you need to install the `acl` package.

```
sudo apt install acl
```

Step 3: Create Apache Virtual Host or Nginx Config File for PostfixAdmin

Apache

If you use Apache web server, create a virtual host for PostfixAdmin.

```
sudo nano /etc/apache2/sites-available/postfixadmin.conf
```

Put the following text into the file. Replace `postfixadmin.example.com` with your real domain name and don't forget to set DNS A record for it.

```
<VirtualHost *:80>

  ServerName postfixadmin.example.com

  DocumentRoot /usr/share/postfixadmin/public



  ErrorLog ${APACHE_LOG_DIR}/postfixadmin_error.log

  CustomLog ${APACHE_LOG_DIR}/postfixadmin_access.log combined



  <Directory />

    Options FollowSymLinks

    AllowOverride All

  </Directory>
```

```
  <Directory /usr/share/postfixadmin/>

    Options FollowSymLinks MultiViews

    AllowOverride All

    Order allow,deny

    allow from all

  </Directory>



</VirtualHost>
```

Save and close the file. Then enable this virtual host with:

```
sudo a2ensite postfixadmin.conf
```

Reload Apache for the changes to take effect.

```
sudo systemctl reload apache2
```

Now you should be able to see the PostfixAdmin web-based install wizard at `http://postfixadmin.example.com/setup.php`.

Nginx

If you use Nginx web server, create a virtual host for PostfixAdmin.

```
sudo nano /etc/nginx/conf.d/postfixadmin.conf
```

Put the following text into the file. Replace `postfixadmin.example.com` with your real domain name and don't forget to set DNS A record for it.

```
server {

   listen 80;

   listen [::]:80;

   server_name postfixadmin.example.com;



   root /usr/share/postfixadmin/public/;

   index index.php index.html;



   access_log /var/log/nginx/postfixadmin_access.log;

   error_log /var/log/nginx/postfixadmin_error.log;



   location / {

       try_files $uri $uri/ /index.php;

   }
```

```
    location ~ ^/(.+\.php)$ {
        try_files $uri =404;
        fastcgi_pass unix:/run/php/php7.4-fpm.sock;
        fastcgi_index index.php;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        include /etc/nginx/fastcgi_params;
    }
}
```

Save and close the file. Then test Nginx configuration.

```
sudo nginx -t
```

If the test is successful, reload Nginx for the changes to take effect.

```
sudo systemctl reload nginx
```

Now you should be able to see the PostfixAdmin web-based install wizard at `http://postfixadmin.example.com/setup.php`.

Step 4: Install Required and Recommended PHP Modules

Run the following command to install PHP modules required or recommended by PostfixAdmin.

```
sudo apt install php7.4-fpm php7.4-imap php7.4-mbstring php7.4-mysql php7.4-json php7.4-curl php7.4-zip
php7.4-xml php7.4-bz2 php7.4-intl php7.4-gmp
```

Then restart Apache. (If you use Nginx, you don't need to restart Nginx.)

```
sudo systemctl restart apache2
```

Step 5: Enabling HTTPS

To encrypt the HTTP traffic, we can enable HTTPS by installing a free TLS certificate issued from Let's Encrypt. Run the following command to install Let's Encrypt client (certbot) on Ubuntu 20.04 server.

```
sudo apt install certbot
```

If you use Apache, install the Certbot Apache plugin.

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

And run this command to obtain and install TLS certificate.

```
sudo certbot --apache --agree-tos --redirect --hsts --staple-ocsp --email you@example.com -d
postfixadmin.example.com
```

If you use Nginx, then you also need to install the Certbot Nginx plugin.

```
sudo apt install python3-certbot-nginx
```

Next, run the following command to obtain and install TLS certificate.

```
sudo certbot --nginx --agree-tos --redirect --hsts --staple-ocsp --email you@example.com -d
postfixadmin.example.com
```

Where

- `--nginx`: Use the nginx plugin.
- `--apache`: Use the Apache plugin.
- `--agree-tos`: Agree to terms of service.
- `--redirect`: Force HTTPS by 301 redirect.
- `--hsts`: Add the Strict-Transport-Security header to every HTTP response. Forcing browser to always use TLS for the domain. Defends against SSL/TLS Stripping.
- `--staple-ocsp`: Enables OCSP Stapling. A valid OCSP response is stapled to the certificate that the server offers during TLS.

The certificate should now be obtained and automatically installed, which is indicated by the message below.

```
IMPORTANT NOTES:
 - Congratulations! Your certificate and chain have been saved at:
   /etc/letsencrypt/live/postfixadmin.linuxbabe.com/fullchain.pem
   Your key file has been saved at:
   /etc/letsencrypt/live/postfixadmin.linuxbabe.com/privkey.pem
   Your cert will expire on 2020-04-14. To obtain a new or tweaked
   version of this certificate in the future, simply run certbot again
   with the "certonly" option. To non-interactively renew *all* of
   your certificates, run "certbot renew"
 - If you like Certbot, please consider supporting our work by:

   Donating to ISRG / Let's Encrypt:   https://letsencrypt.org/donate
   Donating to EFF:                    https://eff.org/donate-le
```

Step 6: Use Strong Password Scheme in PostfixAdmin and Dovecot

By default, PostfixAdmin and Dovecot use MD5-CRYPT, which is a weak password scheme. You can list available password schemes in Dovecot with the following command.

```
sudo doveadm pw -l
```

Sample output:

```
SHA1 SSHA512 BLF-CRYPT PLAIN HMAC-MD5 OTP SHA512 SHA RPA DES-CRYPT CRYPT SSHA MD5-CRYPT SKEY PLAIN-MD4
PLAIN-MD5 SCRAM-SHA-1 LANMAN SHA512-CRYPT CLEAR CLEARTEXT ARGON2I ARGON2ID SSHA256 NTLM MD5 PBKDF2 SHA256
CRAM-MD5 PLAIN-TRUNC SHA256-CRYPT SMD5 DIGEST-MD5 LDAP-MD5
```

Argon2 is a fairly strong password scheme. To use it, we need to edit the PostfixAdmin configuration file, which by default is `/usr/share/postfixadmin/config.inc.php`, but we can create a separate file (`config.local.php`) to store our modifications, so they won't be overwritten when a new version of PostfixAdmin is installed in the future.

```
sudo nano /usr/share/postfixadmin/config.local.php
```

Add the following lines in the file to use Argon2 password scheme.

```
<?php

$CONF['encrypt'] = 'dovecot:ARGON2I';




$CONF['dovecotpw'] = "/usr/bin/doveadm pw -r 5";

if(@file_exists('/usr/bin/doveadm')) { // @ to silence openbase_dir stuff; see
https://github.com/postfixadmin/postfixadmin/issues/171

    $CONF['dovecotpw'] = "/usr/bin/doveadm pw -r 5"; # debian

}
```

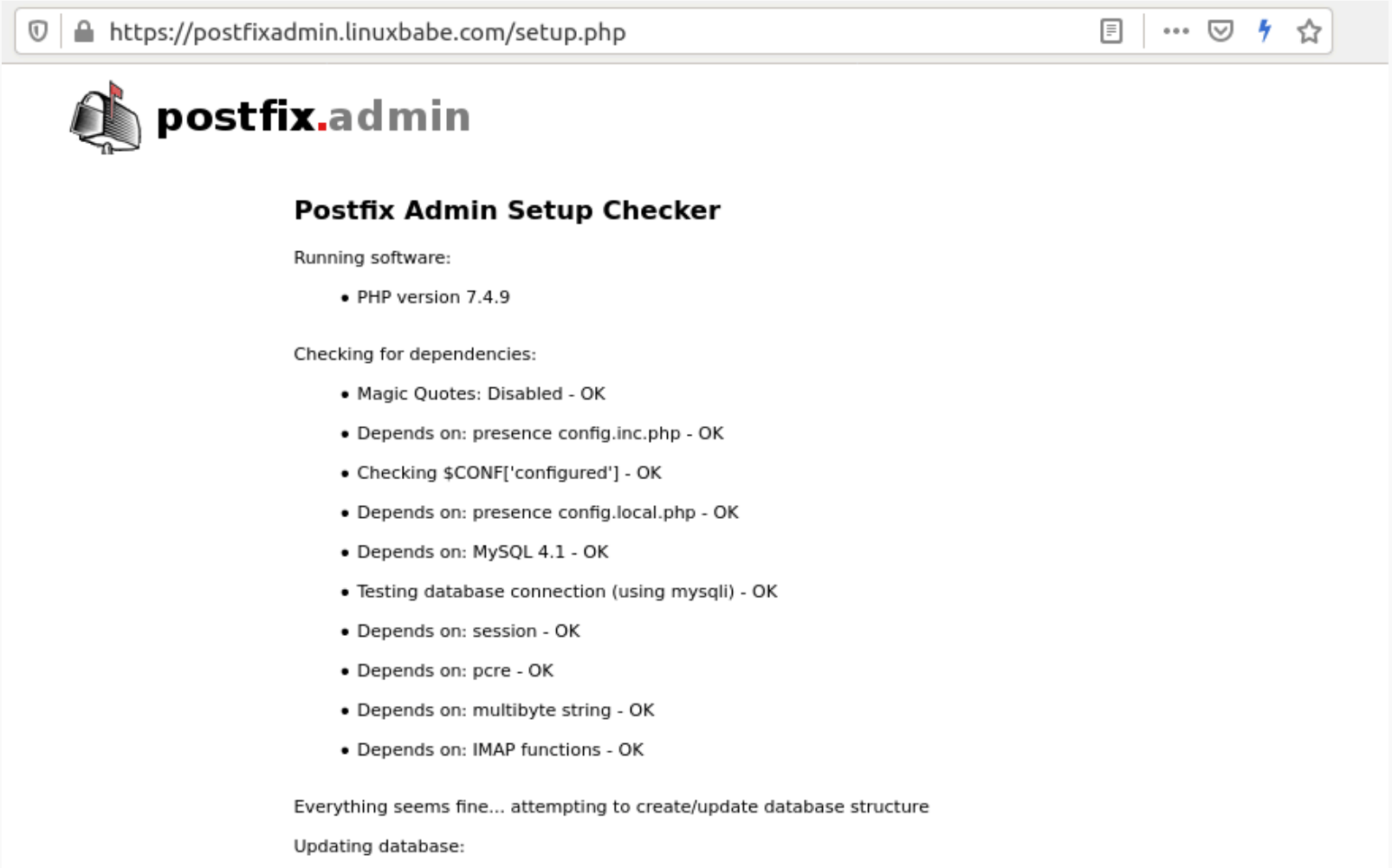
Save and close the file. We can also create a symlink in the `/etc/postfixadmin/` directory, just in case PostfixAdmin can't find the file.

```
sudo ln -s /usr/share/postfixadmin/config.local.php /etc/postfixadmin/config.local.php
```

We will configure password scheme for Dovecot in step 10.

Step 7: Finish the Installation in Web Browser

Go to `postfixadmin.example.com/setup.php` to run the web-based setup wizard. First, it will check if all dependencies are installed.



If you see the following error,

```
Invalid query: Specified key was too long; max key length is 1000 bytes
```

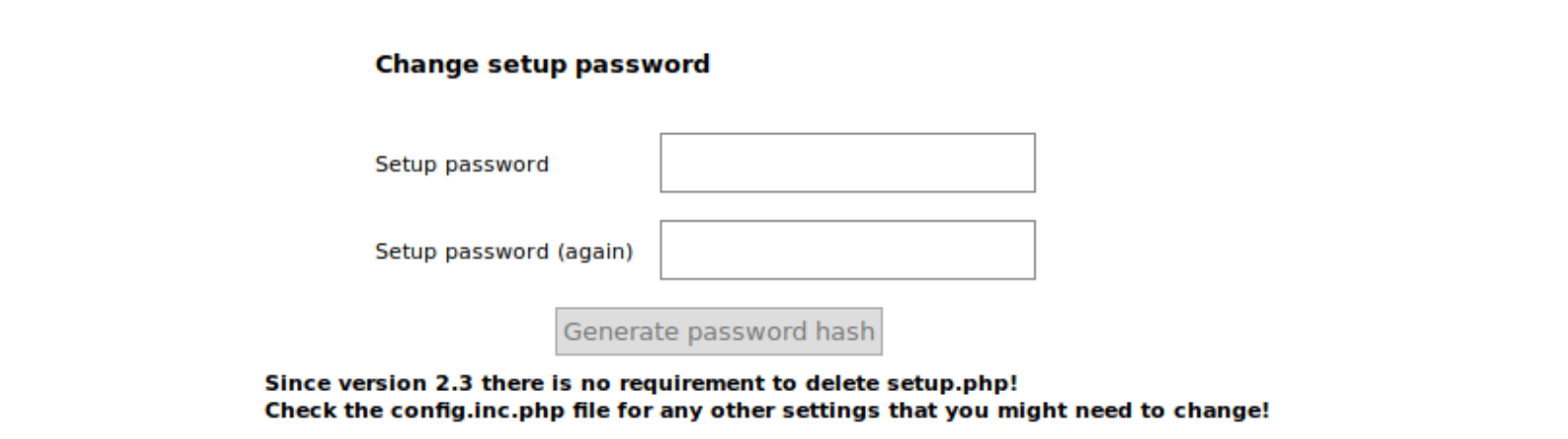
Then you need to log in to MySQL/MariaDB database server as root from command line,

```
sudo mysql -u root
```

and change the default collation from `utf8mb4_general_ci` to `utf8_general_ci`.

```
MariaDB [(none)]> alter database postfixadmin collate ='utf8_general_ci';
```

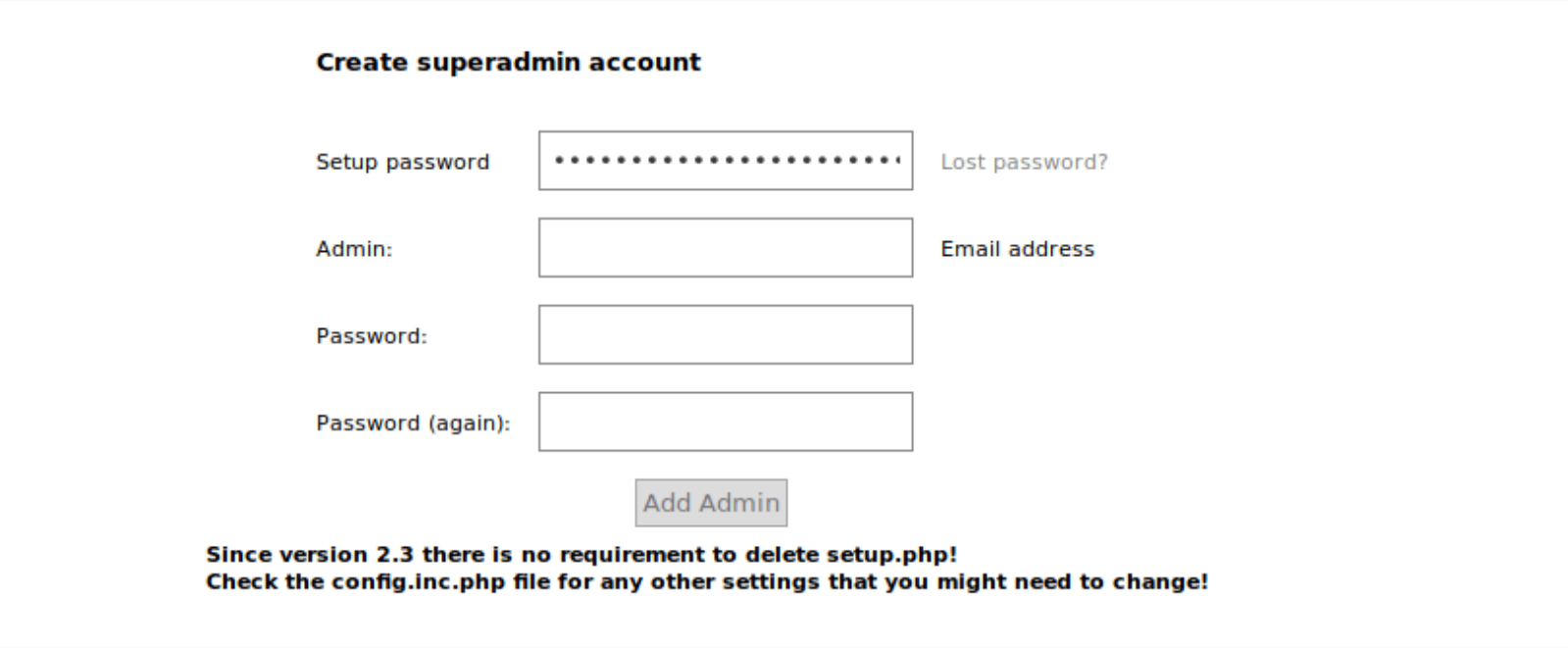
Exit MySQL/MariaDB console and reload the setup.php page. Once all requirements are satisfied, you can create a setup password for PostfixAdmin.



After creating the password hash, you need to open the `/usr/share/postfixadmin/config.local.php` file and add the setup password hash at the end of the file like below. Of course, you need to use your own password hash.

```
<?php

$CONF['encrypt'] = 'dovecot:ARGON2I';
$CONF['dovecotpw'] = "/usr/bin/doveadm pw -r 5";
if(@file_exists('/usr/bin/doveadm')) { // @ to silence openbase_dir stuff; see https://github.com/postfixadmin/pos
tfixadmin/issues/171
    $CONF['dovecotpw'] = "/usr/bin/doveadm pw -r 5"; # debian
}

$CONF['setup_password'] = 'c8b5ddasdfawere1cb15:2a055d9ceed4657faghh3f4349597jmuk67cfafa';
```

Next, create the admin account.



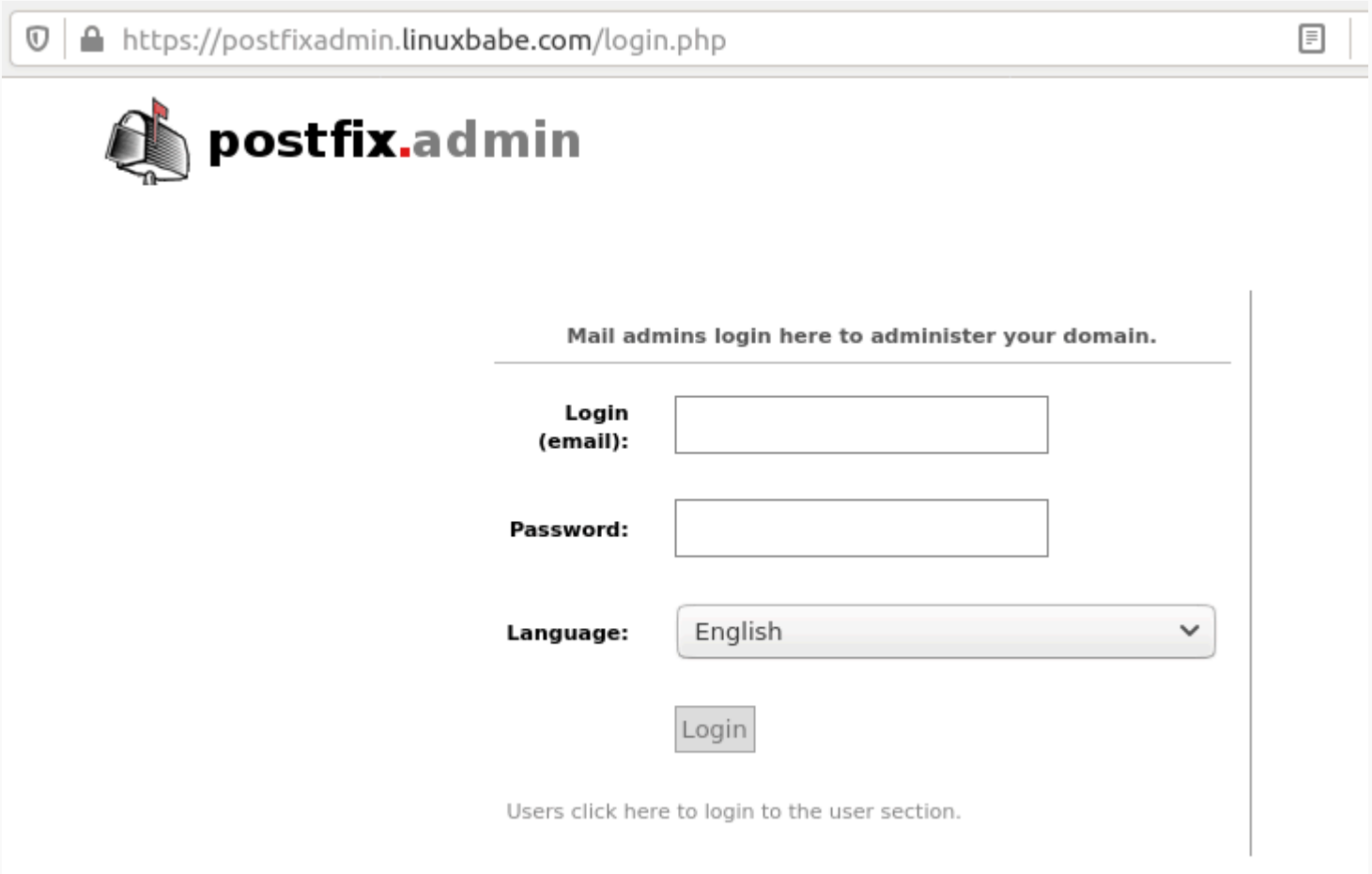
If you see the following error when trying to create a superadmin account,

```
can’t encrypt password with dovecotpw, see error log for details
```

It’s because the `www-data` user doesn’t have permission to read Let’s Encrypt TLS certificate. To fix it, run the following command to grant permissions.

```
sudo setfacl -R -m u:www-data:rx /etc/letsencrypt/live/ /etc/letsencrypt/archive/
```

Once the superadmin account is created, you can log into PostfixAdmin at `postfixadmin.example.com/login.php`.



Step 8: Checking Tables in the Database

The PostfixAdmin setup process populates the `postfixadmin` database with some default tables. It's helpful for us to know the names and structure of the tables. Log in to MySQL/MariaDB console.

```
sudo mysql -u root
```

Select the `postfixadmin` database.

```
USE postfixadmin;
```

List all tables in this database.

```
SHOW TABLES;
```

Output:

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

| Tables_in_postfixadmin |

+------------------------+

| admin                  |

| alias                  |

| alias_domain           |

| config                 |

| domain                 |

| domain_admins          |

| fetchmail              |

| log                    |

| mailbox                |

| quota                  |

| quota2                 |
```

```
| vacation                |

| vacation_notification   |

+-------------------------+

13 rows in set (0.001 sec)
```

The 3 most important tables are:

- `domain`: contains information on the domains that are using your mail server to send and receive email.
- `mailbox`: contains information on every email address, including hashed password and the location of mail files.
- `alias`: contains the alias of each email address.

If you are interested, you can check what columns each table contains. For example, the following command will show us the columns in the `domain` table.

```
DESCRIBE domain;
```

Output:

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

| Field       | Type         | Null | Key | Default             | Extra |

+-------------+--------------+------+-----+---------------------+-------+

| domain      | varchar(255) | NO   | PRI | NULL                |       |

| description | varchar(255) | NO   |     | NULL                |       |
```

```
| aliases     | int(10)      | NO   |     | 0                   |       |

| mailboxes   | int(10)      | NO   |     | 0                   |       |

| maxquota    | bigint(20)   | NO   |     | 0                   |       |

| quota       | bigint(20)   | NO   |     | 0                   |       |

| transport   | varchar(255) | NO   |     | NULL                |       |

| backupmx    | tinyint(1)   | NO   |     | 0                   |       |

| created     | datetime     | NO   |     | 2000-01-01 00:00:00 |       |

| modified    | datetime     | NO   |     | 2000-01-01 00:00:00 |       |

| active      | tinyint(1)   | NO   |     | 1                   |       |

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

Log out of MySQL/MariaDB console.

```
EXIT;
```

Step 9: Configure Postfix to Use MySQL/MariaDB Database

By default, Postfix delivers emails only to users with a local Unix account. To make it deliver emails to virtual users whose information is stored in the database, we need to configure Postfix to use virtual mailbox domains.

First, we need to add MySQL map support for Postfix by installing the `postfix-mysql` package.

```
sudo apt install postfix-mysql
```

Then edit the Postfix main configuration file.

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

Add the following lines at the end of this file.

```
virtual_mailbox_domains = proxy:mysql:/etc/postfix/sql/mysql_virtual_domains_maps.cf

virtual_mailbox_maps =

   proxy:mysql:/etc/postfix/sql/mysql_virtual_mailbox_maps.cf,

   proxy:mysql:/etc/postfix/sql/mysql_virtual_alias_domain_mailbox_maps.cf

virtual_alias_maps =

   proxy:mysql:/etc/postfix/sql/mysql_virtual_alias_maps.cf,

   proxy:mysql:/etc/postfix/sql/mysql_virtual_alias_domain_maps.cf,

   proxy:mysql:/etc/postfix/sql/mysql_virtual_alias_domain_catchall_maps.cf
```

Where:

- `virtual_mailbox_domains` points to a file that will tell Postfix how to look up domain information from the database.
- `virtual_mailbox_maps` points to files that will tell Postfix how to look up email addresses from the database.
- `virtual_alias_maps` points to files that will tell Postfix how to look up aliases from the database.

We want to use dovecot to deliver incoming emails to the virtual users' message store, so also add the following line at the end of this file.

```
virtual_transport = lmtp:unix:private/dovecot-lmtp
```

```
mailbox_transport = lmtp:unix:private/dovecot-lmtp
smtputf8_enable = no

virtual_mailbox_domains = proxy:mysql:/etc/postfix/sql/mysql_virtual_domains_maps.cf
virtual_mailbox_maps =
   proxy:mysql:/etc/postfix/sql/mysql_virtual_mailbox_maps.cf,
   proxy:mysql:/etc/postfix/sql/mysql_virtual_alias_domain_mailbox_maps.cf
virtual_alias_maps =
   proxy:mysql:/etc/postfix/sql/mysql_virtual_alias_maps.cf,
   proxy:mysql:/etc/postfix/sql/mysql_virtual_alias_domain_maps.cf,
   proxy:mysql:/etc/postfix/sql/mysql_virtual_alias_domain_catchall_maps.cf

virtual_transport = lmtp:unix:private/dovecot-lmtp
```

Save and close the file. Next, we need to create the `.cf` files one by one. Create the sql directory.

```
sudo mkdir /etc/postfix/sql/
```

Create the *mysql_virtual_domains_maps.cf* file.

```
sudo nano /etc/postfix/sql/mysql_virtual_domains_maps.cf
```

Add the following content. Replace `password` with the postfixadmin password you set in Step 2.

```
user = postfixadmin

password = password

hosts = localhost
```

```
dbname = postfixadmin

query = SELECT domain FROM domain WHERE domain='%s' AND active = '1'

#query = SELECT domain FROM domain WHERE domain='%s'

#optional query to use when relaying for backup MX

#query = SELECT domain FROM domain WHERE domain='%s' AND backupmx = '0' AND active = '1'

#expansion_limit = 100
```

Create the *mysql_virtual_mailbox_maps.cf* file.

```
sudo nano /etc/postfix/sql/mysql_virtual_mailbox_maps.cf
```

Add the following content.

```
user = postfixadmin

password = password

hosts = localhost

dbname = postfixadmin

query = SELECT maildir FROM mailbox WHERE username='%s' AND active = '1'

#expansion_limit = 100
```

Create the *mysql_virtual_alias_domain_mailbox_maps.cf* file.

```
sudo nano /etc/postfix/sql/mysql_virtual_alias_domain_mailbox_maps.cf
```

Add the following content.

```
user = postfixadmin

password = password

hosts = localhost

dbname = postfixadmin

query = SELECT maildir FROM mailbox,alias_domain WHERE alias_domain.alias_domain = '%d' and
mailbox.username = CONCAT('%u', '@', alias_domain.target_domain) AND mailbox.active = 1 AND
alias_domain.active='1'
```

Create the *mysql_virtual_alias_maps.cf* file.

```
sudo nano /etc/postfix/sql/mysql_virtual_alias_maps.cf
```

Add the following content.

```
user = postfixadmin

password = password

hosts = localhost

dbname = postfixadmin

query = SELECT goto FROM alias WHERE address='%s' AND active = '1'
```

```
#expansion_limit = 100
```

Create the `mysql_virtual_alias_domain_maps.cf` file.

```
sudo nano /etc/postfix/sql/mysql_virtual_alias_domain_maps.cf
```

Add the following content.

```
user = postfixadmin

password = password

hosts = localhost

dbname = postfixadmin

query = SELECT goto FROM alias,alias_domain WHERE alias_domain.alias_domain = '%d' and alias.address =
CONCAT('%u', '@', alias_domain.target_domain) AND alias.active = 1 AND alias_domain.active='1'
```

Create the `mysql_virtual_alias_domain_catchall_maps` file.

```
sudo nano /etc/postfix/sql/mysql_virtual_alias_domain_catchall_maps.cf
```

Add the following content.

```
# handles catch-all settings of target-domain

user = postfixadmin

password = password

hosts = localhost
```

```
dbname = postfixadmin

query = SELECT goto FROM alias,alias_domain WHERE alias_domain.alias_domain = '%d' and alias.address =
CONCAT('@', alias_domain.target_domain) AND alias.active = 1 AND alias_domain.active='1'
```

Since the database passwords are stored in plain text so they should be readable only by user postfix and root, which is done by executing the following two commands.

```
sudo chmod 0640 /etc/postfix/sql/*

sudo setfacl -R -m u:postfix:rx /etc/postfix/sql/
```

Next, we need to change the value of the `mydestination` parameter in Postfix. Display the current value:

```
postconf mydestination
```

Sample output:

```
mydestination = $myhostname, linuxbabe.com, localhost.$mydomain, localhost
```

The `mydestination` parameter contains a list of domain names that will receive emails delivered to local Unix accounts. In part 1, we added the apex domain name (like linuxbabe.com) to `mydestination`. Since we are going to use virtual mailbox, we need to remove the apex domain name from the list by issuing the following command.

```
sudo postconf -e "mydestination = \$myhostname, localhost.\$mydomain, localhost"
```

Now let’s open the Postfix main configuration file again.

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

Add the following lines at the end of this file.

```
virtual_mailbox_base = /var/vmail
```

```
virtual_minimum_uid = 2000

virtual_uid_maps = static:2000

virtual_gid_maps = static:2000
```

The first line defines the base location of mail files. The remaining 3 lines define which user ID and group ID Postfix will use when delivering incoming emails to the mailbox. We use the user ID 2000 and group ID 2000.

Save and close the file. Restart Postfix for the changes to take effect.

```
sudo systemctl restart postfix
```

Next, we need to create a user named `vmail` with ID 2000 and a group with ID 2000.

```
sudo adduser vmail --system --group --uid 2000 --disabled-login --no-create-home
```

Create the mail base location.

```
sudo mkdir /var/vmail/
```

Make `vmail` as the owner.

```
sudo chown vmail:vmail /var/vmail/ -R
```

Step 10: Configure Dovecot to Use MySQL/MariaDB Database

We also need to configure the Dovecot IMAP server to query user information from the database. First, run the following command to add MySQL support for Dovecot.

```
sudo apt install dovecot-mysql
```

Then edit the *10-mail.conf* file.

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

In part 2, we used the following `mail_location`. Email messages are stored under the `Maildir` directory under each user's home directory.

```
mail_location = maildir:~/Maildir
```

Since we are using virtual mailbox domain now, we need to enable `mail_home` for the virtual users by adding the following line in the file, because virtual users don't have home directories by default.

```
mail_home = /var/vmail/%d/%n/
```

```
#
# See doc/wiki/Variables.txt for full list. Some examples:
#
#   mail_location = maildir:~/Maildir
#   mail_location = mbox:~/mail:INBOX=/var/mail/%u
#   mail_location = mbox:/var/mail/%d/%1n/%n:INDEX=/var/indexes/%d/%1n/%n
#
# <doc/wiki/MailLocation.txt>
#
mail_location = maildir:~/Maildir
mail_home = /var/vmail/%d/%n

# If you need to set multiple mailbox locations or want to change default
# namespace settings, you can do it by defining namespace sections.
#
```

Save and close the file. Then edit the *10-auth.conf* file.

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

In part 2, we used the following value for `auth_username_format`.

```
auth_username_format = %n
```

The `%n` would drop the domain if it was given. Because in part 2 we were using local Unix account for the username of every email address, we must use `%n` to drop the domain, so users were able to login with the full email address.

Now we are using virtual mailbox domains, which means the username of every email address includes the domain part, so we need to change the `auth_username_format` as follows. `%u` won't drop away the domain. This allows users to login with the full email address.

```
auth_username_format = %u
```

Uncomment the following line at the end of this file, so Dovecot can query user information from MySQL/MariaDB database.

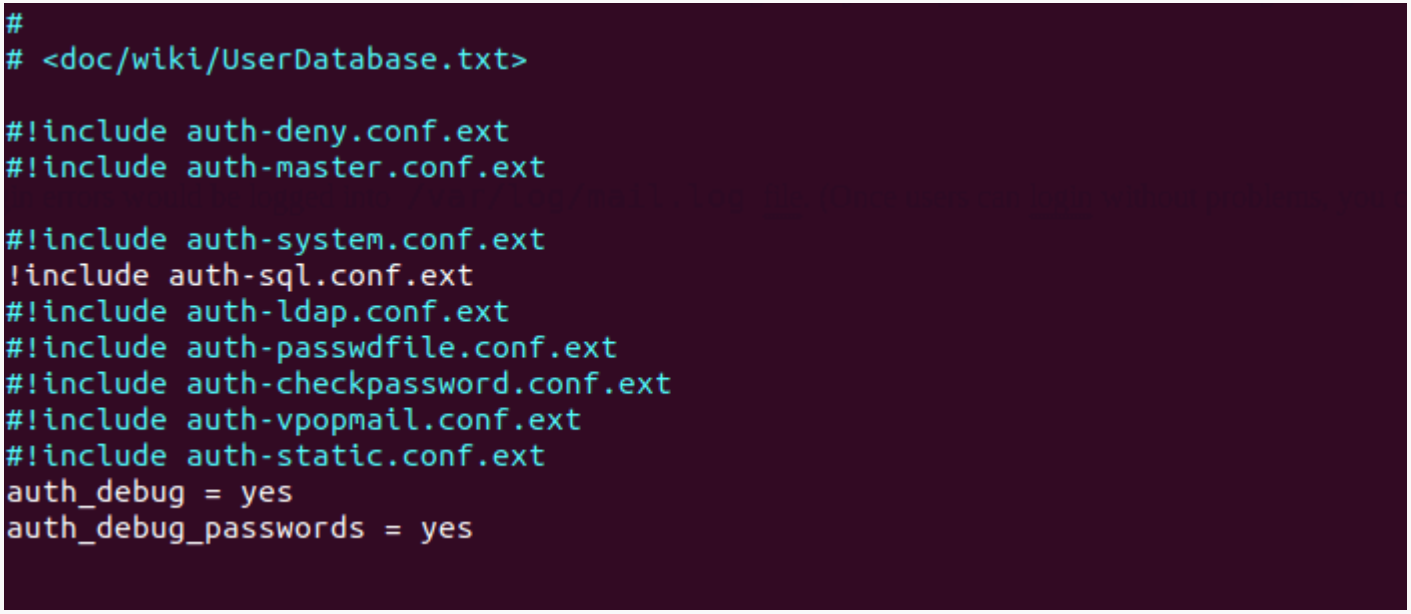
```
!include auth-sql.conf.ext
```

Now you probably don't want local Unix users to send emails without registering email addresses in PostfixAdmin, then comment out the following line by adding the # character at the beginning, so Dovecot won't query the local `/etc/passwd` or `/etc/shadow` file.

```
#!include auth-system.conf.ext
```

It can be helpful to add the following two lines in this file to debug login issues. The login errors would be logged into `/var/log/mail.log` file. (Once users can login without problems, you can comment out the following two lines.)

```
auth_debug = yes

auth_debug_passwords = yes
```



Save and close the file.

Edit the `dovecot-sql.conf.ext` file.

```
sudo nano /etc/dovecot/dovecot-sql.conf.ext
```

Here is the content that you should have in this file. By default, all lines in this file are commented out, so you can simply copy and paste them at the bottom. Replace `password` with the postfixadmin password you set in Step 2.

```
driver = mysql



connect = host=localhost dbname=postfixadmin user=postfixadmin password=password



default_pass_scheme = ARGON2I



password_query = SELECT username AS user,password FROM mailbox WHERE username = '%u' AND active='1'



user_query = SELECT maildir, 2000 AS uid, 2000 AS gid FROM mailbox WHERE username = '%u' AND active='1'



iterate_query = SELECT username AS user FROM mailbox
```

Restart Dovecot.

```
sudo systemctl restart dovecot
```

When a user tries to log in, Dovecot would use the Argon2 algorithm to generate a password hash from the password entered by the user, then compare it with the password hash stored in the database.

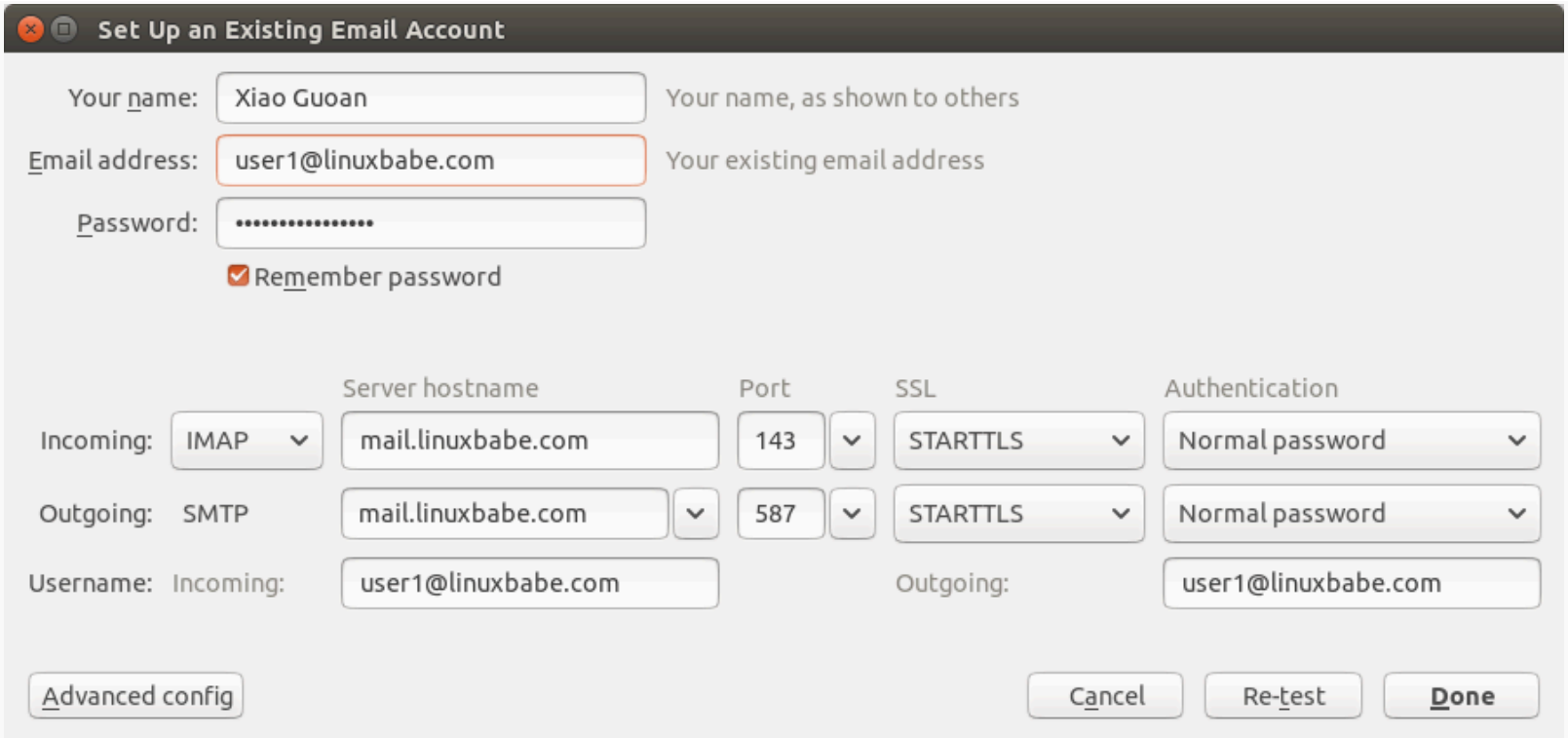
Step 11: Add Domain and Mailboxes in PostfixAdmin

Log in to PostfixAdmin web interface as the admin. Click the `Domain List` tab and select `New Domain` to add a domain. You can choose how many aliases and mailboxes are allowed for this domain.

Then click `Virtual List` tab and select `Add Mailbox` to add a new email address for your domain.

Next, you can open your desktop email client such as Mozilla Thunderbird and add a mail account.

- In the incoming server section, select IMAP protocol, enter `mail.your-domain.com` as the server name, choose port 143 and STARTTLS. Choose `normal password` as the authentication method.
- In the outgoing section, select SMTP protocol, enter `mail.your-domain.com` as the server name, choose port 587 and STARTTLS. Choose `normal password` as the authentication method.



Hint: You can also use port 993 with SSL/TLS encryption for IMAP, and use port 465 with SSL/TLS encryption for SMTP. You should **not** use port 25 as the SMTP port in mail clients to submit outgoing emails.

You should now be able to connect to your own email server and also send and receive emails with your desktop email client! Note that you cannot use local Unix accounts to login now. You must log in with the virtual user created from PostfixAdmin web interface.

Troubleshooting Tips

As a rule of thumb, you should always check the mail log (`/var/log/mail.log`) on your mail server when an error happens. The following is a list of specific errors and troubleshooting tips.

Can’t login from Mail Clients

If you can’t log into your mail server from a desktop mail client, scan your mail server to find if the ports are open. Note that you should run the following command from another Linux computer or server. If you run it on your mail server, then the ports will always appear to be open.

```
sudo nmap mail.your-domain.com
```

And check if Dovecot is running.

```
systemctl status dovecot
```

You can also check the mail log (`/var/log/mail.log`), which may give you some clues. If Dovecot fails to start, the error might not be logged to the `/var/log/mail.log` file, you can run the following command to see what's wrong.

```
sudo journalctl -eu dovecot
```

If you see the following error in the mail log, it's likely that you didn't set a correct password in the `.cf` files under `/etc/postfix/sql/` directory.

```
postfix/trivial-rewrite[28494]: warning: virtual_alias_domains:
proxy:mysql:/etc/postfix/sql/mysql_virtual_alias_maps.cf: table lookup problem

postfix/trivial-rewrite[28494]: warning: virtual_alias_domains lookup failure
```

If you see the following error in the mail log, it's because you forgot to add `mail_location = maildir:~/Maildir` in the `/etc/dovecot/conf.d/10-mail.conf` file.

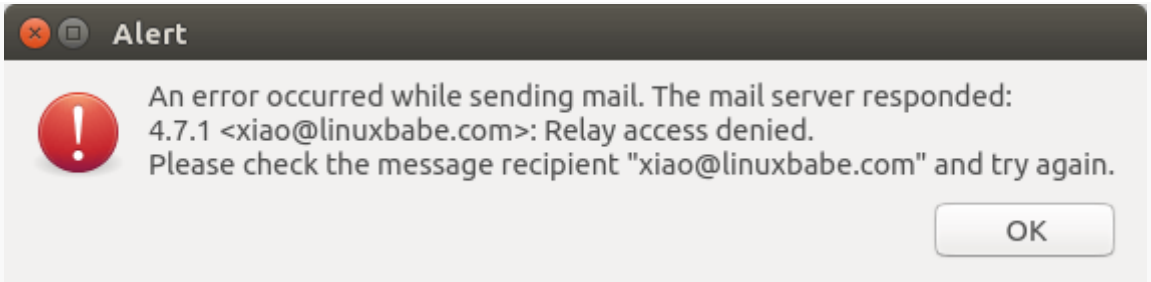
```
open(/var/mail/username@domain.com) failed: Permission denied (euid=2000(vmail) egid=2000(vmail) missing
+w perm: /var/mail, we're not in group 8(mail), dir owned by 0:8 mode=0775
```

Cloudflare DNS

As I said in part 1, if you use Cloudflare DNS service, you should not enable the CDN (proxy) feature when creating DNS A record and AAAA record for the hostname of your mail server. Cloudflare doesn't support SMTP or IMAP proxy.

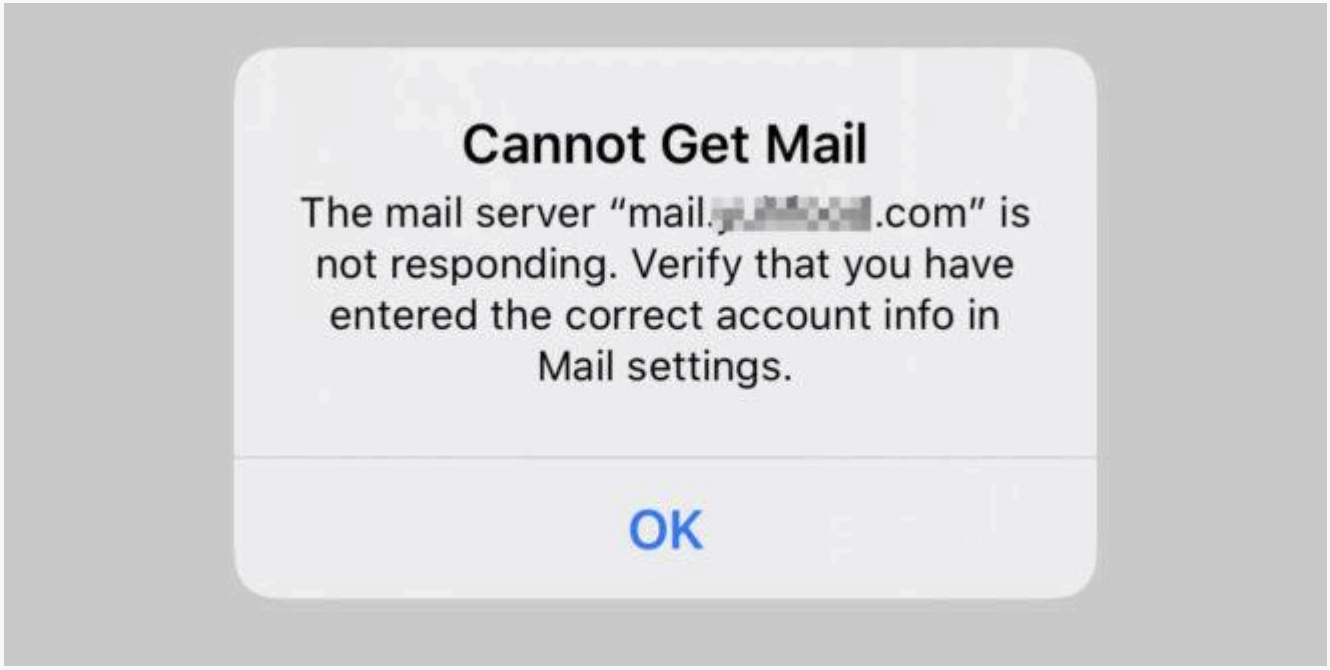
Relay Access Denied

If you see the "**relay access denied**" error when trying to send emails from a mail client, it's most likely that you use port 25 as the SMTP port in your mail client. As I said a while ago, you should use port **587** or **465** as the SMTP port in mail clients (Mozilla Thunberbird, Microsoft Outlook, etc) to submit outgoing emails. Port 25 should be used for SMTP server to SMTP server communications.

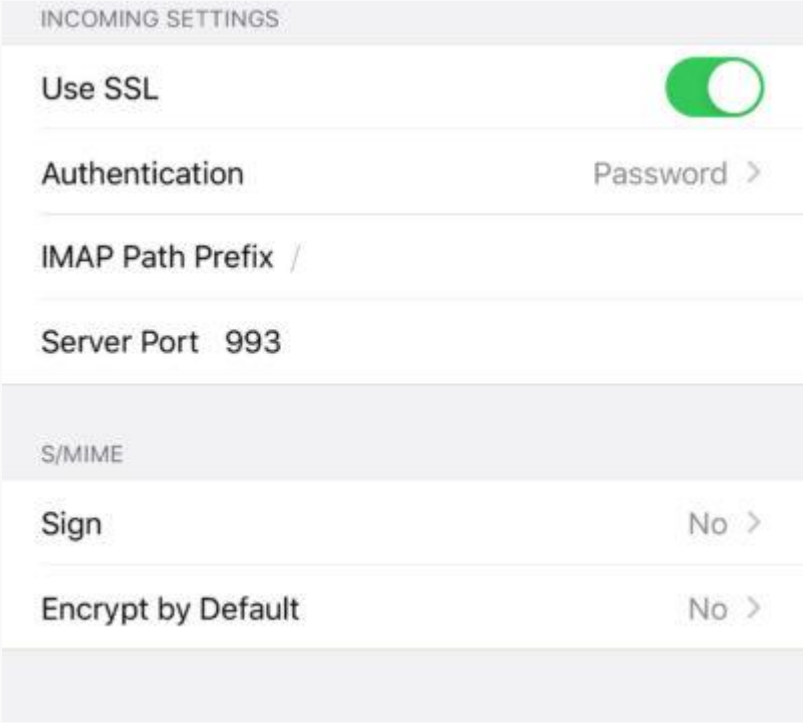


iOS Mail App

If you use the iOS Mail app to log into your mail server and encounter the following error.



You can try to fix it by enforcing SSL encryption, for both SMTP and IMAP.



Fun fact: It seems the iOS Mail app has difficulty in supporting STARTTLS on IMAP port 143, but it supports STARTTLS on the submission port 587.

Automatically Clean the Junk Folder and Trash Folder

To delete emails in Junk folder for all users, you can run

```
sudo doveadm expunge -A mailbox Junk all
```

To delete emails in Trash folder, run

```
sudo doveadm expunge -A mailbox Trash all
```

I think it’s better to clean emails that have been in the Junk or Trash folder for more than 2 weeks, instead of cleaning all emails.

```
sudo doveadm expunge -A mailbox Junk savedbefore 2w
```

Then add a cron job to automate the job.

```
sudo crontab -e
```

Add the following line to clean Junk and Trash folder every day.

```
@daily doveadm expunge -A mailbox Junk savedbefore 2w;doveadm expunge -A mailbox Trash savedbefore 2w
```

To receive report when a Cron job produces an error, you can add the following line above all Cron jobs.

```
MAILTO="you@your-domain.com"
```

Save and close the file. And you're done.

Change User Password in PostfixAdmin

Users can log into PostfixAdmin at `https://postfixadmin.example.com/users/login.php`, then change their passwords.

Restricting Access to Sendmail

By default, any local user can use the `sendmail` binary to submit outgoing emails. Now that your mail server is using virtual mailboxes, you might want to restrict access to the `sendmail` binary to trusted local users only, so a malicious user can't use it to send a large volume of emails to damage your mail server's reputation. Edit the Postfix main configuration file.

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

Add the following line to the end of this file, so only the root and www-data user can submit emails via sendmail. You can also add other usernames.

```
authorized_submit_users = root,www-data
```

Save and close the file. Then restart Postfix.

```
sudo systemctl restart postfix
```