Reference: <https://devanswers.co/configure-postfix-to-use-gmail-smtp-on-ubuntu-16-04-digitalocean-droplet/>

If you’ve already installed Postfix, skip to step 2.

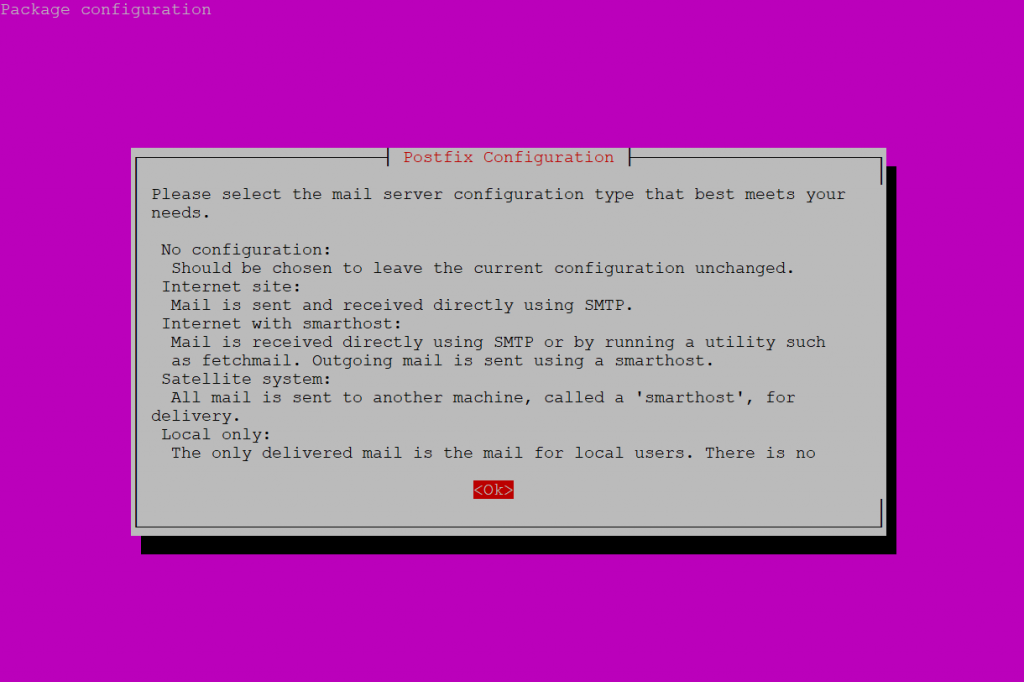
Let’s update the package database first.

sudo apt-get update

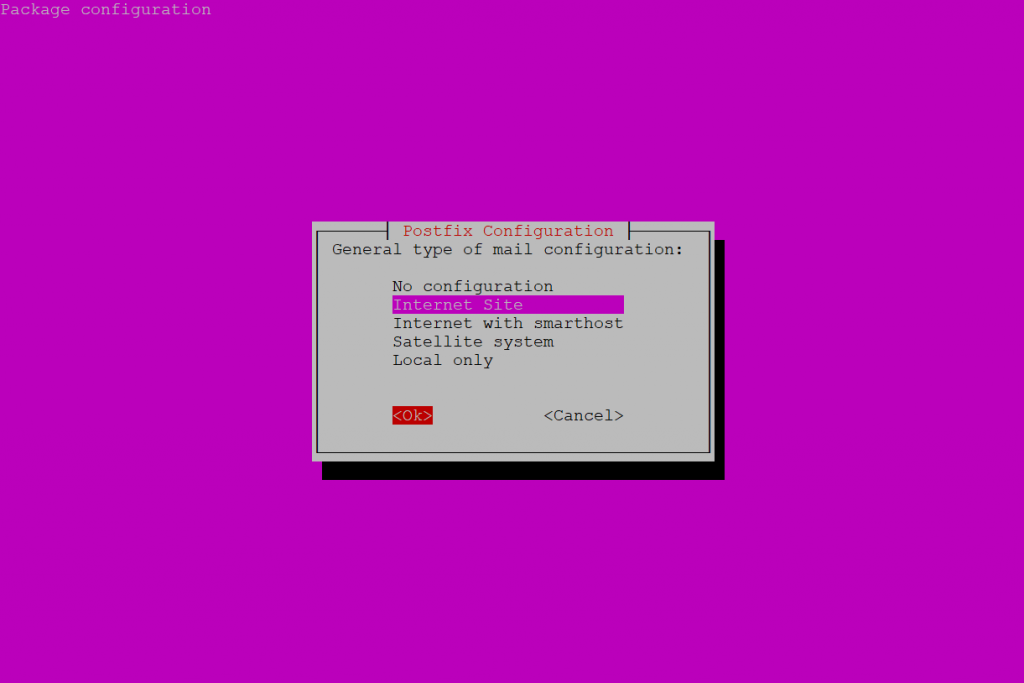
Install mailutils, which will automatically install Postfix.

sudo apt install -y mailutils

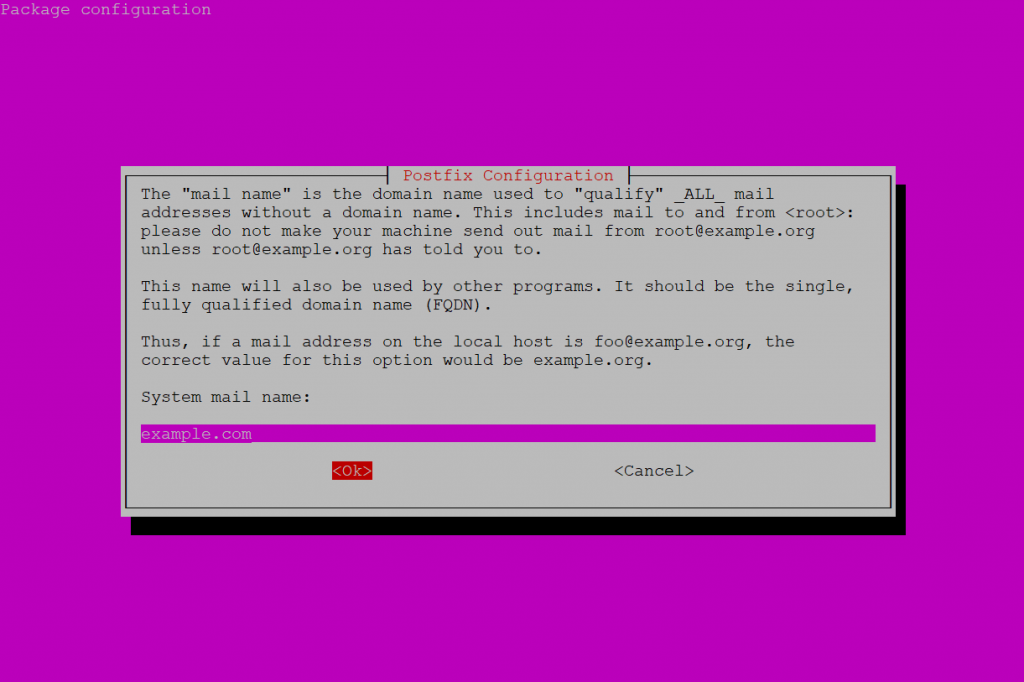
On the first Postfix configuration screen, select **OK** by pressing TAB and ENTER



Select **Internet Site** and press ENTER.



**System mail name** should be your domain name eg. example.com, press ENTER.



Package should now be installed.

**2. Configure Postfix**

Edit the Postfix configuration file.

sudo nano /etc/postfix/main.cf

Find the following line relayhost = about 6 lines up from the bottom of the file and delete it.

Add the following to the end of the file.

/etc/postfix/main.cf

relayhost = [smtp.gmail.com]:587

smtp\_sasl\_auth\_enable = yes

smtp\_sasl\_password\_maps = hash:/etc/postfix/sasl\_passwd

smtp\_sasl\_security\_options = noanonymous

smtp\_tls\_CAfile = /etc/postfix/cacert.pem

smtp\_use\_tls = yes

Save file and exit. (Press CTRL + X, press Y and then press ENTER)

**3. Create Password and DB Files**

Create the sasl\_passwd file which will store our credentials.

sudo nano /etc/postfix/sasl\_passwd

Insert the following:

/etc/postfix/sasl\_passwd

[smtp.gmail.com]:587 username@gmail.com:password

Replace username and password with your own.

Save file and exit. (Press CTRL + X, press Y and then press ENTER)

Create a hash database file for Postfix with the postmap command.

sudo postmap /etc/postfix/sasl\_passwd

There should now be a file called sasl\_passwd.db in the /etc/postfix/ directory.

For added security, we will only allow **root** user to read and write to sasl\_passwd and sasl\_passwd.db

sudo chown root:root /etc/postfix/sasl\_passwd /etc/postfix/sasl\_passwd.db

sudo chmod 0600 /etc/postfix/sasl\_passwd /etc/postfix/sasl\_passwd.db

**4. Sign Certificate**

Now we are going to create the certificate.

cat /etc/ssl/certs/thawte\_Primary\_Root\_CA.pem | sudo tee -a /etc/postfix/cacert.pem

There should now be a certificate file called cacert.pem in /etc/postfix

**5. Send a Test Mail**

We’ll now send a test email message. Make sure to replace test@example.com with your own email address.

echo "Test Email message body" | mail -s "Email test subject" test@example.com

Don’t forget to check your spam folder.

If you still haven’t received any mail, check the mail error log.

sudo tail /var/log/mail.log

If the mail log is empty or doesn’t exist, try parsing the syslog. This will return the last 50 entries for postfix.

sudo tail -f -n 50 /var/log/syslog | grep postfix

If the syslog is empty and you still haven’t received any test email, it’s possible that the test email was rejected by the recipient server. You should check to see if anything has bounced back to your mail folder.

sudo less /var/mail/$(whoami)

Press uppercase G to scroll to the bottom of the file and lowercase q to quit. The $(whoami) variable returns the currently logged in user.

**6. Allow Less Secure Apps**

If Gmail is not allowing postfix to connect via SMTP, you may need to enable “Allow Less Secure Apps” on your Gmail account.

Please see: [Allow less secure apps to access your Gmail account](https://devanswers.co/allow-less-secure-apps-access-gmail-account/)

You need to allow less secure apps, you can do it by click below link

<https://www.google.com/settings/security/lesssecureapps>

or

<https://accounts.google.com/DisplayUnlockCaptcha>

**Configure authentication SASL**

<https://www.palawandigital.com/configuring-postfix-encryption-ubuntu/>

# Installing and Configuring Postfix on an Ubuntu 16.04 Web Server

In this guide, we’ll see how to install, configure and enable TLS encryption and SMTP authentication on Postfix, a free mail transfer agent for Ubuntu. This will allow your web server to send email notifications (for WordPress for example) and to do it in a way that respect the encryption standards.

This guide is for Ubuntu 16.04, but note that this tutorial should also perfectly work on an Ubuntu 14.04 web server.

## 1. Install Postfix

1. Run this command to start the Postfix installation:

*apt-get install postfix*

2. Choose “Internet Site”

3. Enter the FQDN of the server

## 2. Edit the Aliases

Edit this file:

*nano /etc/aliases*

Add the missing “root” line:

*# See man 5 aliases for format*

*postmaster: root*

*root: admin@example.com*

For more informations:

* [Digital Ocean](https://www.digitalocean.com/community/tutorials/how-to-install-and-configure-postfix-as-a-send-only-smtp-server-on-ubuntu-14-04#step-4-%E2%80%94-forward-system-mail)
* [Tecmint](http://www.tecmint.com/install-wordpress-lamp-postfix-mail-notifications-vps-server/)
* [Postfix](http://www.postfix.org/documentation.html)

## 3. Reconfigure Postfix

To ensure everything is set properly, we’ll run this command:

*sudo dpkg-reconfigure postfix*

If not done already, change the System Domain name to your FQDN:

*hostname.domain.com*

In the “Root and postmaster mail recipient” field, add the name of the main admin user:

*your\_admin\_user\_name*

Other destinations for mail:

*server1.example.com, example.com, localhost.example.com, localhost*

Force synchronous updates on mail queue? Answer: No

Leave the local networks as is.

Mailbox size limit (bytes): 0

Local address extension character: +

Internet protocols to use: all

## 4. Optional: Configure the mailbox format (Advanced Users Only)

If you are not sure about this, just skip it and go directly to step 5.

To configure the mailbox format for Maildir:

*sudo postconf -e 'home\_mailbox = Maildir/'*

You may need to issue this as well:

*sudo postconf -e 'mailbox\_command ='*

Note: This will place new mail in /home/username/Maildir so you will need to configure your Mail Delivery Agent to use the same path.

## 5. Enable TLS Encryption For Postfix

Configure Postfix to do SMTP AUTH using SASL (saslauthd):

*sudo postconf -e 'smtpd\_sasl\_local\_domain ='*

*sudo postconf -e 'smtpd\_sasl\_auth\_enable = yes'*

*sudo postconf -e 'smtpd\_sasl\_security\_options = noanonymous'*

*sudo postconf -e 'broken\_sasl\_auth\_clients = yes'*

*sudo postconf -e 'smtpd\_recipient\_restrictions = permit\_sasl\_authenticated,permit\_mynetworks,reject\_unauth\_destination'*

*sudo postconf -e 'inet\_interfaces = all'*

Create or edit this file:

*nano /etc/postfix/sasl/smtpd.conf*

Add the following lines:

*pwcheck\_method: saslauthd*

*mech\_list: plain login*

## 6. Generate certificates to be used for TLS encryption and/or certificate Authentication

*touch smtpd.key*

*chmod 600 smtpd.key*

*openssl genrsa 1024 > smtpd.key*

*openssl req -new -key smtpd.key -x509 -days 3650 -out smtpd.crt*

*openssl req -new -x509 -extensions v3\_ca -keyout cakey.pem -out cacert.pem -days 3650*

*sudo mv smtpd.key /etc/ssl/private/*

*sudo mv smtpd.crt /etc/ssl/certs/*

*sudo mv cakey.pem /etc/ssl/private/*

*sudo mv cacert.pem /etc/ssl/certs/*

## 7. Configure Postfix to do TLS encryption for both incoming and outgoing mail

*sudo postconf -e 'smtp\_tls\_security\_level = may'*

*sudo postconf -e 'smtpd\_tls\_security\_level = may'*

*sudo postconf -e 'smtpd\_tls\_auth\_only = no'*

*sudo postconf -e 'smtp\_tls\_note\_starttls\_offer = yes'*

*sudo postconf -e 'smtpd\_tls\_key\_file = /etc/ssl/private/smtpd.key'*

*sudo postconf -e 'smtpd\_tls\_cert\_file = /etc/ssl/certs/smtpd.crt'*

*sudo postconf -e 'smtpd\_tls\_CAfile = /etc/ssl/certs/cacert.pem'*

*sudo postconf -e 'smtpd\_tls\_loglevel = 1'*

*sudo postconf -e 'smtpd\_tls\_received\_header = yes'*

*sudo postconf -e 'smtpd\_tls\_session\_cache\_timeout = 3600s'*

*sudo postconf -e 'tls\_random\_source = dev:/dev/urandom'*

Remember to change this to yours:

*sudo postconf -e 'myhostname = server1.example.com'*

## 8. Restart the postfix daemon like this

*sudo /etc/init.d/postfix restart*

## 9. Configure Postfix To Use SASL For SMTP AUTH

Install libsasl2-2, sasl2-bin and libsasl2-modules

*apt-get install libsasl2-2 sasl2-bin libsasl2-modules*

Edit this file in order to activate saslauthd:

*nano /etc/default/saslauthd*

Remove # in front of START=yes, add the PWDIR, PARAMS, and PIDFILE lines and edit the OPTIONS line at the end:

*# This needs to be uncommented before saslauthd will be run automatically*

*START=yes*

*PWDIR="/var/spool/postfix/var/run/saslauthd"*

*PARAMS="-m ${PWDIR}"*

*PIDFILE="${PWDIR}/saslauthd.pid"*

*# You must specify the authentication mechanisms you wish to use.*

*# This defaults to "pam" for PAM support, but may also include*

*# "shadow" or "sasldb", like this:*

*# MECHANISMS="pam shadow"*

*MECHANISMS="pam"*

*# Other options (default: -c)*

*# See the saslauthd man page for information about these options.*

*#*

*# Example for postfix users: "-c -m /var/spool/postfix/var/run/saslauthd"*

*# Note: See /usr/share/doc/sasl2-bin/README.Debian*

*#OPTIONS="-c"*

*#make sure you set the options here otherwise it ignores params above and will not work*

*OPTIONS="-c -m /var/spool/postfix/var/run/saslauthd"*

We then update the dpkg “state” of /var/spool/postfix/var/run/saslauthd. The saslauthd init script uses this setting to create the missing directory with the appropriate permissions and ownership:

*sudo dpkg-statoverride --force --update --add root sasl 755 /var/spool/postfix/var/run/saslauthd*

**Note**: This may report an error that “–update given” and the “/var/spool/postfix/var/run/saslauthd” directory does not exist. You can ignore this because when you start saslauthd next it will be created.

Apparently, the saslauthd looks for the config file /etc/saslauthd and not for /etc/default/saslauthd. This link fixes this issue:

*sudo ln -s /etc/default/saslauthd /etc/saslauthd*

## 10. Finally, start saslauthd

*sudo /etc/init.d/saslauthd start*

To see if SMTP-AUTH and TLS work properly now run the following command:

*telnet localhost 25*

After you have established the connection to your postfix mail server type:

*ehlo localhost*

If you see the lines:

*250-STARTTLS*

*250-AUTH*

…among others, everything is working. Type quit to return to the system’s shell.

Your server should now send encrypted messages. If you still get a warning from your email provider, consider adding an SPF text to your DNS.

**Configure FTP**

<https://www.digitalocean.com/community/tutorials/how-to-set-up-vsftpd-for-a-user-s-directory-on-ubuntu-18-04>

**Step 1 — Installing vsftpd**

Let’s start by updating our package list and installing the vsftpd daemon:

sudo apt update

sudo apt install vsftpd

When the installation is complete, let’s copy the configuration file so we can start with a blank configuration, saving the original as a backup:

sudo cp /etc/vsftpd.conf /etc/vsftpd.conf.orig

With a backup of the configuration in place, we’re ready to configure the firewall.

**Step 2 — Opening the Firewall**

Let’s check the firewall status to see if it’s enabled. If it is, we’ll ensure that FTP traffic is permitted so firewall rules don’t block our tests.

Check the firewall status:

sudo ufw status

In this case, only SSH is allowed through:

Output

Status: active

To Action From

-- ------ ----

OpenSSH ALLOW Anywhere

OpenSSH (v6) ALLOW Anywhere (v6)

You may have other rules in place or no firewall rules at all. Since only SSH traffic is permitted in this case, we’ll need to add rules for FTP traffic.

Let’s open ports 20 and 21 for FTP, port 990 for when we enable TLS, and ports 40000-50000 for the range of passive ports we plan to set in the configuration file:

sudo ufw allow 20/tcp

sudo ufw allow 21/tcp

sudo ufw allow 990/tcp

sudo ufw allow 40000:50000/tcp

sudo ufw status

Our firewall rules should now look like this:

Output

Status: active

To Action From

-- ------ ----

OpenSSH ALLOW Anywhere

990/tcp ALLOW Anywhere

20/tcp ALLOW Anywhere

21/tcp ALLOW Anywhere

40000:50000/tcp ALLOW Anywhere

OpenSSH (v6) ALLOW Anywhere (v6)

20/tcp (v6) ALLOW Anywhere (v6)

21/tcp (v6) ALLOW Anywhere (v6)

990/tcp (v6) ALLOW Anywhere (v6)

40000:50000/tcp (v6) ALLOW Anywhere (v6)

With vsftpd installed and the necessary ports open, let’s move on to creating a dedicated FTP user.

**Step 3 — Preparing the User Directory**

We will create a dedicated FTP user, but you may already have a user in need of FTP access. We’ll take care to preserve an existing user’s access to their data in the instructions that follow. Even so, we recommend that you start with a new user until you’ve configured and tested your setup.

First, add a test user:

sudo adduser sammy

Assign a password when prompted. Feel free to press ENTER through the other prompts.

FTP is generally more secure when users are restricted to a specific directory. vsftpd accomplishes this with [chroot](https://www.digitalocean.com/community/tutorials/how-to-configure-chroot-environments-for-testing-on-an-ubuntu-12-04-vps" \l "what-is-a-chroot-environment) jails. When chroot is enabled for local users, they are restricted to their home directory by default. However, because of the way vsftpd secures the directory, it must not be writable by the user. This is fine for a new user who should only connect via FTP, but an existing user may need to write to their home folder if they also have shell access.

In this example, rather than removing write privileges from the home directory, let’s create an ftp directory to serve as the chroot and a writable files directory to hold the actual files.

Create the ftp folder:

sudo mkdir /home/sammy/ftp

Set its ownership:

sudo chown nobody:nogroup /home/sammy/ftp

Remove write permissions:

sudo chmod a-w /home/sammy/ftp

sudo chmod a+w /home/sammy/ftp #make root writeable

Verify the permissions:

sudo ls -la /home/sammy/ftp

Output

total 8

4 dr-xr-xr-x 2 nobody nogroup 4096 Aug 24 21:29 .

4 drwxr-xr-x 3 sammy sammy 4096 Aug 24 21:29 ..

Next, let’s create the directory for file uploads and assign ownership to the user:

sudo mkdir /home/sammy/ftp/files

sudo chown sammy:sammy /home/sammy/ftp/files

A permissions check on the ftp directory should return the following:

sudo ls -la /home/sammy/ftp

Output

total 12

dr-xr-xr-x 3 nobody nogroup 4096 Aug 26 14:01 .

drwxr-xr-x 3 sammy sammy 4096 Aug 26 13:59 ..

drwxr-xr-x 2 sammy sammy 4096 Aug 26 14:01 files

Finally, let’s add a test.txt file to use when we test:

* echo "vsftpd test file" | sudo tee /home/sammy/ftp/files/test.txt

Now that we’ve secured the ftp directory and allowed the user access to the files directory, let’s modify our configuration.

**Step 4 — Configuring FTP Access**

We’re planning to allow a single user with a local shell account to connect with FTP. The two key settings for this are already set in vsftpd.conf. Start by opening the config file to verify that the settings in your configuration match those below:

sudo nano /etc/vsftpd.conf

/etc/vsftpd.conf

. . .

# Allow anonymous FTP? (Disabled by default).

anonymous\_enable=NO

#

# Uncomment this to allow local users to log in.

local\_enable=YES

. . .

Next, let’s enable the user to upload files by uncommenting the write\_enable setting:

/etc/vsftpd.conf

. . .

write\_enable=YES

. . .

We’ll also uncomment the chroot to prevent the FTP-connected user from accessing any files or commands outside the directory tree:

/etc/vsftpd.conf

. . .

chroot\_local\_user=YES

. . .

Let’s also add a user\_sub\_token to insert the username in our local\_root directory path so our configuration will work for this user and any additional future users. Add these settings anywhere in the file:

/etc/vsftpd.conf

. . .

user\_sub\_token=$USER

local\_root=/home/$USER/ftp

Let’s also limit the range of ports that can be used for passive FTP to make sure enough connections are available:

/etc/vsftpd.conf

. . .

pasv\_min\_port=40000

pasv\_max\_port=50000

**Note:** In step 2, we opened the ports that we set here for the passive port range. If you change the values, be sure to update your firewall settings.

To allow FTP access on a case-by-case basis, let’s set the configuration so that users have access only when they are explicitly added to a list, rather than by default:

/etc/vsftpd.conf

. . .

userlist\_enable=YES

userlist\_file=/etc/vsftpd.userlist

userlist\_deny=NO

**pam\_service\_name=ftp**

allow\_writeable\_chroot=YES

userlist\_deny toggles the logic: When it is set to YES, users on the list are denied FTP access. When it is set to NO, only users on the list are allowed access.

When you’re done making the changes, save the file and exit the editor.

Finally, let’s add our user to /etc/vsftpd.userlist. Use the -a flag to append to the file:

echo "sammy" | sudo tee -a /etc/vsftpd.userlist

Check that it was added as you expected:

cat /etc/vsftpd.userlist

Output

sammy

Restart the daemon to load the configuration changes:

sudo systemctl restart vsftpd

With the configuration in place, let’s move on to testing FTP access.

Add user:

Useradd $username

Passwd $username