Postfix as Outgoing SMTP Mail Server Ubuntu 18.04 / Debian 9.9 Stretch Setup

Postfix is a Mail Transfer Agent (MTA), this means that it enables the transfer of emails from your system to another following the RFC standards for mail transfer. In a simple view, Postfix understands everything there is about sending emails, but not necessarily much about receiving them.

Run the following to get started, which will install the latest version of postfix on your system:

```
sudo apt-get update
sudo apt-get install postfix -y
```

Now if you want to *only* send mail and not receive them, using your favorite editor, open the config file at `/etc/postfix/main.cf` , and find the following line:

```
inet_interfaces = all
```

and change it to

```
inet_interfaces = your-domain.com , localhost
```

Otherwise don't edit that line. Now restart Postfix using the following command:

```
sudo systemctl restart postfix
```

Setup OpenDKIM for Postfix

DKIM is a system that validates that the mail was signed by a particular entity and hasn’t been tampered with.

It tells you the location of the domain which signed the email, but does not say that it was the domain that sent it.

The Authentication-Results header says dkim=pass so the DKIM signature was validated successfully. DKIM doesn’t make an assertion that the signing domain has anything to do with the sender in the ‘from’ header, which is why later on, we have to have add other records to our DNS settings for the domain to verify the origin and permitted sender. In summary, DKIM consists of a private and public key, where the mail is signed by the private key before being sent and the receiving domain/server (outlook.com, Gmail) will contact the DNS registrar to receive information about the public key, then verify the mail has not been modified since being sent. It is *not* an encryption scheme, merely a check to make sure that the mail has not been modified.

```
sudo apt-get install opendkim opendkim-tools -y

sudo adduser postfix opendkim
```

And edit the following lines in `/etc/opendkim.conf` from

```
#Canonicalization simple

#Mode sv

#SubDomains no
```

To

```
Canonicalization relaxed/simple
Mode s
SubDomains no
```

Then append the following at the end of `/etc/opendkim.conf`

```
# Map domains in From addresses to keys used to sign messages
KeyTable refile:/etc/opendkim/key.table
SigningTable refile:/etc/opendkim/signing.table
# A set of internal hosts whose mail should be signed
InternalHosts /etc/opendkim/trusted.hosts
```

These files do not exist yet, and hence we need to create them. To create the signing-table, key-table and trusted hosts files, enter the following:

```
sudo mkdir -p /etc/opendkim/keys
sudo chown -R opendkim:opendkim /etc/opendkim
sudo chmod go-rw /etc/opendkim/keys
sudo touch /etc/opendkim/signing.table
sudo touch /etc/opendkim/key.table
sudo touch /etc/opendkim/trusted.hosts
```

This will also give opendkim permissions over these files. We'll begin with editing the signing table file at `/etc/opendkim/signing.table` . Change the contents of the file to the following:

```
*@example.com sendonly._domainkey.example.com
```

Any more domains you want to send on behalf of need to be added on a new line. Now we move on to editing the `key.table` file at `/etc/opendkim/key.table` , and fill the contents with the following:

```
sendonly._domainkey.example.com example.com:sendonly:/etc/opendkim/keys/exampl
e.com/sendonly.private
```

Once again, if you want to send on behalf of multiple domains, you need to add each domain on a separate line, replacing `example.com` with the domain you want.

Save and exit that file and finally open the `trusted.hosts` file at `/etc/opendkim/trusted.hosts` and fill the contents with:

```
127.0.0.1
localhost


*.example.com
```

Any more domains you are sending on behalf of need to be added on a new line with the format `*.newdomain.com`

Generate Private/Public Keypair

Previously when creating the `key.table` file, we edited the contents to point to a key located at `/etc/opendkim/keys/exmaple.com/sendonly.private` but these key does not yet exist. We need to generate a private key to sign outgoing emails and a public key for receiving SMTP servers to verify the DKIM signature. THe public key will be published in the DNS settings for your domain(s). Perform the following for each domain you wish to send on behalf of:

Create a separate folder for the domain you want to generate keys for (replacing `example.com` with your domain):

```
sudo mkdir /etc/opendkim/keys/example.com
```

Then generate the keys for each domain `using`:

```
sudo opendkim-genkey -b 2048 -d example.com -D /etc/opendkim/keys/example.com -
s sendonly -v
sudo chown opendkim:opendkim /etc/opendkim/keys/example.com/sendonly.private
```

Repeat the above process for each domain you want to have, replacing `example.com` with your domains.

Note that `sendonly` is still the selector here. This will be the prefix of `_domainkey` in our DNS record. Now display the public key and make sure to add it to the DNS record:

```
sudo cat /etc/opendkim/keys/example.com/sendonly.txt
```

This will print an unholy amount of unnecessary information as our domain provider namecheap manages the formatting for us. It gives an output like the following:

```
sendonly._domainkey IN TXT ( "v=DKIM1; h=sha256; k=rsa; "
      "p=MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAwVVC8RGSynFIN18JekVJlstZif
u1BZG85y8F4Ir/IJ5Uvmjk3kTG0fFpckCogWFUYKtWonnDpokdJ2RiH5xZGQ56/C6D6Ms3wkkuL4n47
2DkJLXEHwOkv44acF7eA9sBm+lM+T4OHsKmopfmpTf2Kv20WmgCGZO46w+14eRmGWz7yr94OwF6a8Py
xdz5mGheOItnywLHgM8OoT"
      "xkFqwruvVP0X/RKNh/ehDBZRk3fW0I5MD+iHT2+sReNH4jjQRiMp6weVvn3FDo3UdpwKAGZs
eRCdP0ZZ1+W5KJ8usIuyLeVSiOUCH+COAo5sKVergj3UgN8279thgsiX+Wi86QOQIDAQAB" ) ; --
--- DKIM key sendonly for example.com
```

In the above example, we see that we need to set a TXT record in our DNS settings for the domain `example.com` with the host of `sendonly._domainkey` and the value then needs to be formatted like the following:

```
v=DKIM1; h=sha256; k=rsa; p=MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAwVVC8RG
SynFIN18JekVJlstZifu1BZG85y8F4Ir/IJ5Uvmjk3kTG0fFpckCogWFUYKtWonnDpokdJ2RiH5xZGQ
56/C6D6Ms3wkkuL4n472DkJLXEHwOkv44acF7eA9sBm+1M+T4OHsKmopfmpTf2Kv20WmgCGZO46w+14
eRmGWz7yr94OwF6a8Pyxdz5mGheOItnywLHgM8OoTxkFqwruvVP0X/RKNh/ehDBZRk3fW0I5MD+iHT2
+sReNH4jjQRiMp6weVvn3FDo3UdpwKAGZseRCdP0ZZ1+W5KJ8usIuyLeVSiOUCH+COAo5sKVergj3Ug
N8279thgsiX+Wi86QOQIDAQAB
```

Notice how the value of the DNS record was the text between the brackets `( )` but with quotation marks removed and the line breaks removed as well. Once this has been set in the DNS record for the domain, you should be able to test that the key is properly set with the command:

```
sudo opendkim-testkey -d example.com -s sendonly -vvv
```

Note that you may have to wait for the DNS records to be propagated beforehand (set your TTL for the TXT record to 1min if you want it done quickly). If you see `key not secure` message, don't panic. This is because DNSSEC isn't enabled on the domain name.

Connect Postfix with OpenDKIM

Edit the configuration file at `/etc/opendkim.conf` and add this line (if it already exists, replace it with the following):

```
Socket local:/var/spool/postfix/opendkim/opendkim.sock
```

```
sudo mkdir /var/spool/postfix/opendkim
sudo chown opendkim:postfix /var/spool/postfix/opendkim
```

Now, if you can find the following line in `/etc/default/opendkim` file

```
SOCKET="local:/var/run/opendkim/opendkim.sock"
```

Or

```
SOCKET=local:$RUNDIR/opendkim.sock
```

Replace it with

```
SOCKET="local:/var/spool/postfix/opendkim/opendkim.sock"
```

And then we need to edit the main Postfix configuration file located at `/etc/postfix/main.cf` to have the following lines at the end to connect Postfix with OpenDKIM.

```
milter_default_action = accept
milter_protocol = 6
smtpd_milters = local:/opendkim/opendkim.sock
non_smtpd_milters = $smtpd_milters
```

Now restart opendkim and Postfix with

```
sudo systemctl restart opendkim postfix
```

OpenDKIM won't produce any message if it fails to restart. Run the following command to check its status.

Make sure it's running by checking:

```
systemctl status opendkim
```

Finally test an email using the `swaks` tool, first install it with

```
apt install swaks
```

and send an email to yourself using

```
swaks --server localhost --to myself@myemail.com
```

I'd recommend using a Gmail account to test sending to since you can view the source of the email to see the security level of the mail. Notice that it is most likely still going to spam since it's not configured with SSL / TLS.

Before we do that however it's worth one last change to the `/etc/postfix/master.cf` file. Uncomment the sections so that the file looks like the following:

```
# # Postfix master process configuration file. For details on the format
# of the file, see the master(5) manual page (command: "man 5 master" or
# on-line: http://www.postfix.org/master.5.html).
# # Do not forget to execute "postfix reload" after editing this file.
# # ==========================================================================
# service type private unpriv chroot wakeup maxproc command + args
# (yes) (yes) (no) (never) (100)
# ==========================================================================
smtp inet n - y - - smtpd
#smtp inet n - y - 1 postscreen
#smtpd pass - - y - - smtpd
#dnsblog unix - - y - 0 dnsblog
#tlsproxy unix - - y - 0 tlsproxy
#submission inet n - y - - smtpd
  -o syslog_name=postfix/submission
  -o smtpd_tls_security_level=encrypt
  -o smtpd_sasl_auth_enable=yes
# -o smtpd_reject_unlisted_recipient=no
# -o smtpd_client_restrictions=$mua_client_restrictions
# -o smtpd_helo_restrictions=$mua_helo_restrictions
# -o smtpd_sender_restrictions=$mua_sender_restrictions
# -o smtpd_recipient_restrictions=
# -o smtpd_relay_restrictions=permit_sasl_authenticated,reject
# -o milter_macro_daemon_name=ORIGINATING
smtps inet n - y - - smtpd
  -o syslog_name=postfix/smtps
  -o smtpd_tls_wrappermode=yes
  -o smtpd_sasl_auth_enable=yes
# -o smtpd_reject_unlisted_recipient=no
# -o smtpd_client_restrictions=$mua_client_restrictions
# -o smtpd_helo_restrictions=$mua_helo_restrictions
# -o smtpd_sender_restrictions=$mua_sender_restrictions
...
```

This is only the first few lines of the file, but what needs to be changed exists in those first few lines. Notice that

`smtp inet n - y - - smtpd`

was uncommented and then the first three options beginning with `-o` followed by `smtps inet n - y - - smtpd` being uncommented along with the next three launch options with `-o` . This is to allow us to have IMAP and SMTP access from 3rd party mail clients like roundcube or the Gmail app on mobile devices. **Also note that spaces at the beginning of the lines are important so separate from the previous option!**

Configure TLS with Postfix and Certbot

Even after signing the email with DKIM, we don't have any security for the mail, only authenticity. Email is effectively a plaintext communication sent from email clients to receiving email servers or from one server to another. This design limitation leaves the content of a message in transit open for anyone to eavesdrop, anyone sniffing packets that are being transferred by your computer can read the email data in plaintext.

Transport Layer Security (TLS) helps solve this issue by offering encryption technology for your message while it is “in transit” from one secure email server to another. Initially a secure session is estabilished using Diffie Helman key exchange to exchange public keys securely, then email transmission can be established since now the receiving server can decrypt the message upon receiving the mail.

So in short, DKIM has ensured the authenticity of the mail, now TLS will ensure the security of the mail. Install and run certbot using the following commands:

```
sudo apt install certbot -y
sudo certbot certonly --standalone -d example.com
```

Obviously replace `example.com` with the domain you are using.

Now we add our newly created keys to postfix using the commands, replacing `example.com` as necessary:

```
sudo postconf -e 'smtpd_tls_cert_file = /etc/letsencrypt/live/example.com/fullc
hain.pem'
sudo postconf -e 'smtpd_tls_key_file = /etc/letsencrypt/live/example.com/privke
y.pem'
sudo postconf -e 'smtpd_sasl_type = dovecot'
sudo postconf -e 'smtpd_sasl_path = private/auth'
sudo postconf -e 'smtpd_sasl_local_domain ='
sudo postconf -e 'smtpd_sasl_security_options = noanonymous'
sudo postconf -e 'broken_sasl_auth_clients = yes'
sudo postconf -e 'smtpd_sasl_auth_enable = yes'
sudo postconf -e 'smtpd_recipient_restrictions = permit_sasl_authenticated,perm
it_mynetworks,reject_unauth_destination'
```

```
sudo postconf -e 'smtp_tls_security_level = may'
```

```
sudo postconf -e 'smtpd_tls_security_level = may'
```

```
sudo postconf -e 'smtp_tls_note_starttls_offer = yes'
```

```
sudo postconf -e 'smtpd_tls_loglevel = 1'
```

```
sudo postconf -e 'smtpd_tls_received_header = yes'
```

```
sudo postconf -e "home_mailbox = Maildir/"
```

```
sudo systemctl restart postfix
```

Configure Dovecot

Dovecot is mailbox interface software. Specifically, it allows users to access their mailboxes using the IMAP interface. If configured correctly, it only interacts with authenticated users. This means that if we wish to in the future have access to our system mail remotely, we can connect using the IMAP or POP3 interfaces. This is identical to logging into a Gmail or Outlook account on your phone or through the web interface. A simple view would be that Postfix only knows about sending emails, but Dovecot knows about reading emails.

Install Dovecot and its affiliates:

```
sudo apt install dovecot-common dovecot-imapd dovecot-pop3d
```

Then edit the following file at `/etc/dovecot/conf.d/10-auth.conf` to have the lines be (I've used ellipses in the following configurations to signify a gap between sections in the file, do not put these ellipses in any configuration files):

```
disable_plaintext_auth = yes
...
auth_mechanisms = plain login
```

Then instruct the mail directory to use the same format as Postfix. Edit `/etc/dovecot/conf.d/10-mail.conf` to say:

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

Next, configure the IMAP and POP3 protocols for email clients in the `master.conf` file as shown below. Uncomment the port lines shown underneath by deleting '#' sign at the start of these lines. In the same file, also edit the `service auth` segment to allow user authentication. Open `/etc/dovecot/conf.d/10-master.conf` and edit to:

```
service imap-login {
  inet_listener imap {
    port = 143
  } inet_listener imaps {
    port = 993
    ssl = yes
  }
...
} service pop3-login {
  inet_listener pop3 {
    port = 110
  }
  inet_listener pop3s {
    port = 995
    ssl = yes
  }
}
...

service auth {

...

    # Postfix smtp-auth
  unix_listener /var/spool/postfix/private/auth {
    mode = 0666
    user = postfix
    group = postfix
  }
```

And finally, force TLS by editing `/etc/dovecot/conf.d/10-ssl.conf` (don't forget to replace `example.com` with your domain) to say:

```
# SSL/TLS support: yes, no, required. <doc/wiki/SSL.txt>
ssl = required
...
```

```
ssl_cert = </etc/letsencrypt/live/example.com/fullchain.pem

ssl_key = </etc/letsencrypt/live/example.com/privkey.pem


...


# SSL protocols to use

ssl_protocols = !SSLv3
```

Once everything looks correct, restart Dovecot to apply the new settings.

```
sudo systemctl restart dovecot
```

Configure Postfix to send emails with Dovecot

We now need to tell Postfix to work with Dovecot. Edit /etc/postfix/main.cf with:

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

Look for the mailbox_command, delete it, and add the following in the end:

```
mailbox_command = /usr/lib/dovecot/dovecot-lda -f "$SENDER" -a "$RECIPIENT"
```

Look for the mydestination line and replace it with:

```
mydestination = $myhostname, localhost.com, localhost, localhost.localdomain,
localhost.$myhostname
```

That should be it. Now Postfix is installed, and secured with DKIM and TLS. Test your email by using `swaks` again, but instead send to a gmail address since you can view the security component of the message.

Now reload postfix and restart dovecot and postfix:

```
sudo postfix reload
sudo systemctl restart dovecot opendkim postfix
```

Setting a DMARC & SPF DNS record

DMARC & SPF are the final aspects of the security trio “integrity, confidentiality, authenticity”. We will be assuring the authenticity of the sender using DMARC & SPF records. TLS assures confidentiality of the mail and DKIM assures integrity of the mail.

SPF doesn’t care about the From header in the mail. It only cares about the envelope sender (the server). If you send a mail from your server, the Authentication-Results header would show this was something@example.com. If the SPF record passed, then example.com did allow the sending IP (the envelope sender) as per its SPF record.

DMARC is the only thing that cares about the ‘From’ header. It will look at the from header and see if either the DKIM was signed by example.com, or the smtp.mailfrom address was something@example.com. If at least one of those is true, DMARC passes. If neither are true, DMARC fails.

These two in conjunction makes sure that the sender is in fact allowed to send from the domain example.com, verifying the authenticity of the email. Now no one can use another server to send mail and claim that they are sending from your domain.

DMARC DNS Settings

To set the DMARC record, enter the following as the value for a `TXT` record into your DNS settings for your domain:

```
v=DMARC1; p=quarantine
```

With the host as `_dmarc` .

Setting SPF records and how to allow sending as another domain or subdomain

If you want to send email as another domain or subdomain, say example.net or mail.example.com, you'll need to let example.net know that you are going to be sending mail as example.net but from example.com. This is done using an SPF record for example.net that specifies the IP address and hostname that the host of example.com. In the case of linkscircle.com to be the sender for other domains:

Set `TXT` record with host `@` and value:

```
v=spf1 ip4:71.19.249.15 ip4:149.28.175.74 a:mail.example.com include:mail.examp
le.com include:_spf.google.com ~all
```

The above allows mail.example.com to be the sender, where the second ip4 address is the server for mail.example.com. If you don't wish to allow sending as another domain, ignore the `a:mail.example.com` and `include:mail.example.com` and the ip4 entry that corresponds to the `mail.example.com` subdomain. Make sure to leave your actual server `example.com` s ip4 address in though.

Troubleshooting

Are Postfix and Dovecot Running

Sometimes your mail server is not functioning correctly because the needed services are not running. For a mail server that has been running for a long time, resource overuse is the most likely cause of stopped services. It doesn’t hurt to check your resource use to rule out that problem. However, when you’re just setting up a new mail server, it’s more likely that your service startup problems are being caused by configuration errors. Some configuration errors - particularly syntax errors - are serious enough that they can prevent a service from starting.

To check that Postfix and Dovecot are running and to find startup errors, follow these steps:

Run this command to check that Postfix is running:

```
service postfix status
```

You should see the following output:

```
* postfix is running
```

Next, run this command to check that Dovecot is running:

```
service dovecot status
```

You should see output similar to the following:

```
dovecot start/running, process 2241
```

Examine the results. If you see no output, or output that says stop/waiting or not running, the service is not running. The next step is to try restarting the services.

Try to restart the services. Restarting Postfix and Dovecot is also a good troubleshooting procedure even if they’re currently running, because then you can examine the startup messages, which can give you troubleshooting clues. Enter the following command to restart Postfix:

```
 service postfix restart
```

Or reload the configuration:

```
service postfix reload
```

You should see the following messages:

```
* Stopping Postfix Mail Transport Agent postfix                    [ OK ]
* Starting Postfix Mail Transport Agent postfix                    [ OK ]
```

Execute the following command to restart Dovecot:

```
 service dovecot restart
```

You should see the following messages:

```
 dovecot stop/waiting

 dovecot start/running, process 31171
```

1. Examine the results. If you get an error, or the restart message for Dovecot doesn't include a new process ID, there's something preventing the service from starting.
2. If you received a specific error from the restart attempt, search for it online.

Check the applications' startup logs to see more detailed messages. Postfix's stop and start messages are logged in /var/log/mail.log (along with all its other messages) or /var/log/mail.err or /var/log/mail.warn . Enter the following command to view the most recent lines in the log:

```
 sudo tail -100 /var/log/mail.log
```

On a normal restart, you should see the following:

```
Apr 15 16:21:42 godel postfix/master[19624]: terminating on signal 15
Apr 15 16:21:42 godel postfix/master[20232]: daemon started -- version 2.9.6, configuration
/etc/postfix
```

Dovecot’s default startup log is also in /var/log/mail.log. On a normal restart, you should see the following:

```
Apr 15 16:22:45 master: Warning: Killed with signal 15 (by pid=1 uid=0 code=kill)

Apr 15 16:24:12 master: Info: Dovecot v2.0.19 starting up (core dumps disabled)
```

Note: If you moved the Dovecot logs, the normal Dovecot startup messages will be in /var/log/dovecot.log instead. If you can’t find the Dovecot logs, locate them with the following command:

```
doveadm log find
```

If you don’t see these normal startup messages, check for errors instead. Search for errors online.

If there’s a problem during Dovecot’s startup, you should also check /var/log/upstart/dovecot.log. On a normal startup, nothing will be logged to this file. However, if there is a startup problem, an entry will be added in this log which can be quite helpful. To view this file, run the following command:

```
tail /var/log/upstart/dovecot.log
```

Here’s an example where a syntax error in the /etc/dovecot/conf.d/10-master.conf file has been identified:

```
doveconf: Fatal: Error in configuration file /etc/dovecot/conf.d/10-master.conf line 36:
Unexpected '}'
```

If you find a syntax error, open up the offending file and look at the line mentioned (Line 36 in the example above). It’s actually fairly common to get syntax errors during the Dovecot setup process, because there are so many different files and a lot of nested brackets.

Check the Logs

If Postfix, Dovecot, and MySQL are running, the next troubleshooting step is to check the mail logs. By default, all of the incoming and outgoing connections and any associated errors get logged in /var/log/mail.log. One of the most helpful ways to view the log file is with the tail command, which when combined with the -f flag, shows you the most recent part of the log live as it's updated.

Start tailing the log by entering the following command:

```
tail -f /var/log/mail.log
```

1. Send yourself a test message with swaks or make a connection to the mail server.
2. View the log as it updates with the relevant information.
3. To stop tailing, press **CTRL-C**.

If you see an error or warning in the log, copy it. Search for that exact error online (without the details specific to your server), and you'll likely be able to find a solution or additional troubleshooting help.

Enable Verbose Logs

The default mail log may not contain all the information you need. In that case, the next step is to enable verbose logging for Postfix and Dovecot, and to separate the Postfix and Dovecot logs into two separate files so they're easier to sort through. The Postfix log will document messages that are relayed to or from outside servers, and the Dovecot log will record authorization attempts.

Dovecot

Follow these instructions to enable verbose logging for Dovecot and change the log location to /var/log/dovecot.log:

Open the /etc/dovecot/conf.d/10-logging.conf file for editing by entering the following command:

```
nano /etc/dovecot/conf.d/10-logging.conf
```

Add this line to set the new file path for the log:

```
log_path = /var/log/dovecot.log
```

Uncomment the auth_verbose and mail_debug lines, and then set them to yes:

```
auth_verbose = yes

mail_debug = yes
```

Save your changes and restart Dovecot by entering the following command:

```
 service dovecot restart
```

The Dovecot log will now display more information about authorization attempts and inbox connections. You can view the new log at /var/log/dovecot.log. Remember to disable verbose logging when you're done troubleshooting so your server doesn't fill up with logs.

Postfix

Follow these instructions to enable verbose logging for Postfix:

Open the /etc/postfix/master.cf files for editing by entering the following command:

```
nano /etc/postfix/master.cf
```

Add a -v to the smtp line to enable verbose logging:

```
smtp      inet  n       -       -       -       -       smtpd -v
```

Save your changes.

Restart Postfix by entering the following command:

```
service postfix restart
```

The Postfix log will now display more information about messages that are coming from or going to outside servers. You can still view the log at /var/log/mail.log. Remember to disable verbose logging when you're done troubleshooting so your server doesn't fill up with logs.

Check Port Availability

Sometimes email problems occur because the mail server and mail client aren't talking to each other on the same ports. For mail to get from client to server, or vice versa, both have to be using the same ports, and those ports also have to be open along the internet route between the two. If you are following the accompanying Postfix, Dovecot, and MySQL installation guide, you should be using the following ports:

- 25, 465, or 587 with TLS encryption for outgoing mail (SMTP)
- 993 with SSL encryption for incoming IMAP
- 995 with SSL encryption for incoming POP3

First, check your mail client settings and make sure that you have the correct ports and security settings selected.

Next, use the Telnet tool to check that ports are open both on your Linode and on the route between your client and your Linode. The same test should be run on both your Linode and your home computer. First we'll present how to run the test from both locations, and then we'll discuss the implications.

Checking from a Linode

To test on your Linode, follow these steps:

1. Establish an SSH connection to your Linode.

Run the following command, replacing 12.34.56.78 with your Linode's IP address:

```
telnet 12.34.56.78 25
```

2. Exit Telnet by pressing **CTRL-]**, then enter quit.
3. Repeat Step 2 for ports 465, 587, 993, and 995.

Analyzing the Results

If the test is successful, you should see output similar to the following:

```
Trying 12.34.56.78...
Connected to li468-222.members.linode.com.
Escape character is '^]'.
220 host.example.com ESMTP Postfix (Ubuntu)
```

To cancel the connection, press **CTRL-]**, then enter quit. If the test fails, you will see a Connection refused message and Telnet will quit on its own.

If you run the test on your Linode and it fails, you should check that you've configured the ports properly in your mail server setup, that you've enabled ports 465 and 587 (see Steps 26-30 in the Postfix section of the setup guide), and that you don't have any Firewall rules in place that block them.

If you run the test on your Linode and it succeeds, but the test from your home computer fails, that indicates that the ports are being blocked somewhere on the network between your home computer and your Linode. It could be at your router, your ISP (Internet Service Provider), someone else's ISP, etc. The best way to diagnose networking issues is to generate an MTR report.

If the Telnet tests on your Linode and your home computer both succeed, and your mail client settings are correct, you can probably rule out any problems with ports.

Configuration files samples

I attach here an output of the main configuration files discussed in this tutorial while the main hostname is replaced with MYHOST.com

/etc/postfix/main.cf

```
# See /usr/share/postfix/main.cf.dist for a commented, more complete version


# Debian specific:  Specifying a file name will cause the first
# line of that file to be used as the name.  The Debian default
# is /etc/mailname.
#myorigin = /etc/mailname

smtpd_banner = $myhostname ESMTP $mail_name (Debian/GNU)
biff = no

# appending .domain is the MUA's job.
append_dot_mydomain = no
```

```
# Uncomment the next line to generate "delayed mail" warnings
#delay_warning_time = 4h

readme_directory = no

# See http://www.postfix.org/COMPATIBILITY_README.html -- default to 2 on
# fresh installs.
compatibility_level = 2

# TLS parameters
smtpd_tls_cert_file = /etc/letsencrypt/live/MYHOPST.com/fullchain.pem
smtpd_tls_key_file = /etc/letsencrypt/live/MYHOST.com/privkey.pem
smtpd_use_tls = yes
smtpd_tls_session_cache_database = btree:${data_directory}/smtpd_scache
smtp_tls_session_cache_database = btree:${data_directory}/smtp_scache

# See /usr/share/doc/postfix/TLS_README.gz in the postfix-doc package for
# information on enabling SSL in the smtp client.

smtpd_relay_restrictions = permit_mynetworks permit_sasl_authenticated defer_unauth_destination
myhostname = MYHOST.com
alias_maps = hash:/etc/aliases
alias_database = hash:/etc/aliases
myorigin = /etc/mailname
mydestination = $myhostname, localhost.com, localhost, localhost.localdomain, localhost.$myhostname
relayhost =
mynetworks = 127.0.0.0/8 [::ffff:127.0.0.0]/104 [::1]/128
mailbox_size_limit = 0
recipient_delimiter = +
inet_interfaces = MYHOST.com , localhost
inet_protocols = all

milter_default_action = accept
milter_protocol = 6
smtpd_milters = local:/opendkim/opendkim.sock
non_smtpd_milters = $smtpd_milters
smtpd_sasl_type = dovecot
smtpd_sasl_path = private/auth
smtpd_sasl_local_domain =
smtpd_sasl_security_options = noanonymous
broken_sasl_auth_clients = yes
smtpd_sasl_auth_enable = yes
smtpd_recipient_restrictions = permit_sasl_authenticated,permit_mynetworks,reject_unauth_destination
smtp_tls_security_level = may
smtpd_tls_security_level = may
smtp_tls_note_starttls_offer = yes
```

```
smtpd_tls_loglevel = 1
smtpd_tls_protocols = !SSLv2, !SSLv3
smtpd_tls_received_header = yes
home_mailbox = Maildir/
mailbox_command = /usr/lib/dovecot/dovecot-lda -f "$SENDER" -a "$RECIPIENT"
```

/etc/postfix/master.cf

```
#
# Postfix master process configuration file.  For details on the format
# of the file, see the master(5) manual page (command: "man 5 master" or
# on-line: http://www.postfix.org/master.5.html).
#
# Do not forget to execute "postfix reload" after editing this file.
#
# ==========================================================================
# service type  private unpriv  chroot  wakeup  maxproc command + args
#               (yes)   (yes)   (no)    (never) (100)
# ==========================================================================

smtp    inet  n       -       y       -       -       smtpd -v
#smtp      inet  n       -       y       -       1       postscreen
#smtpd     pass  -       -       y       -       -       smtpd
#dnsblog   unix  -       -       y       -       0       dnsblog
#tlsproxy  unix  -       -       y       -       0       tlsproxy
submission inet n       -       y       -       -       smtpd
  -o syslog_name=postfix/smtps
  -o smtpd_tls_security_level=encrypt
  -o smtpd_sasl_auth_enable=yes
#  -o smtpd_reject_unlisted_recipient=no
#  -o smtpd_client_restrictions=$mua_client_restrictions
#  -o smtpd_helo_restrictions=$mua_helo_restrictions
#  -o smtpd_sender_restrictions=$mua_sender_restrictions
#  -o smtpd_recipient_restrictions=
#  -o smtpd_relay_restrictions=permit_sasl_authenticated,reject
#  -o milter_macro_daemon_name=ORIGINATING
smtps  inet  n       -       y       -       -       smtpd
 -o syslog_name=postfix/smtps
 -o smtpd_tls_wrappermode=yes
 -o smtpd_sasl_auth_enable=yes
#  -o smtpd_reject_unlisted_recipient=no
#  -o smtpd_client_restrictions=$mua_client_restrictions
#  -o smtpd_helo_restrictions=$mua_helo_restrictions
#  -o smtpd_sender_restrictions=$mua_sender_restrictions
#  -o smtpd_recipient_restrictions=
#  -o smtpd_relay_restrictions=permit_sasl_authenticated,reject
#  -o milter_macro_daemon_name=ORIGINATING
#628   inet  n        -       y       -       -       qmqpd
pickup  unix  n        -       y       60      1       pickup
cleanup   unix  n      -       y       -       0       cleanup
qmgr    unix  n        -       n       300     1       qmgr
#qmgr   unix  n        -       n       300     1       oqmgr
tlsmgr  unix  -        -       y       1000?   1       tlsmgr
rewrite   unix  -      -       y       -       -       trivial-rewrite
bounce          unix  -        -       y       -       0       bounce
defer   unix  -        -       y       -       0       bounce
trace   unix  -        -       y       -       0       bounce
verify  unix  -        -       y       -       1       verify
flush   unix  n        -       y       1000?   0       flush
proxymap  unix  -      -       n       -       -       proxymap
proxywrite unix -      -       n       -       1       proxymap
smtp    unix  -        -       y       -       -       smtp
relay   unix  -        -       y       -       -       smtp
#       -o smtp_helo_timeout=5 -o smtp_connect_timeout=5
showq unix  n          -       y       -       -       showq
error   unix  -        -       y       -       -       error
retry   unix  -        -       y       -       -       error
discard   unix  -      -       y       -       -       discard
local   unix  -        n       n       -       -       local
virtual   unix  -      n       n       -       -       virtual
lmtp    unix  -        -       y       -       -       lmtp
anvil   unix  -        -       y       -       1       anvil
scache unix  -         -       y       -       1       scache
#
# ==================================================================
# Interfaces to non-Postfix software. Be sure to examine the manual
# pages of the non-Postfix software to find out what options it wants.
#
# Many of the following services use the Postfix pipe(8) delivery
# agent.  See the pipe(8) man page for information about ${recipient}
# and other message envelope options.
# ==================================================================
#
# maildrop. See the Postfix MAILDROP_README file for details.
# Also specify in main.cf: maildrop_destination_recipient_limit=1
#
maildrop  unix  -      n       n       -       -       pipe
  flags=DRhu user=vmail argv=/usr/bin/maildrop -d ${recipient}
#
# ==================================================================
#
# Recent Cyrus versions can use the existing "lmtp" master.cf entry.
```

```
#
# Specify in cyrus.conf:
#   lmtp           cmd="lmtpd -a" listen="localhost:lmtp" proto=tcp4
#
# Specify in main.cf one or more of the following:
#  mailbox_transport = lmtp:inet:localhost
#  virtual_transport = lmtp:inet:localhost
#
# ====================================================================
#
# Cyrus 2.1.5 (Amos Gouaux)
# Also specify in main.cf: cyrus_destination_recipient_limit=1
#
#cyrus unix  -       n       n       -       -       pipe
#  user=cyrus argv=/cyrus/bin/deliver -e -r ${sender} -m ${extension} ${user}
#
# ====================================================================
# Old example of delivery via Cyrus.
#
#old-cyrus unix  -    n       n       -       -       pipe
#  flags=R user=cyrus argv=/cyrus/bin/deliver -e -m ${extension} ${user}
#
# ====================================================================
#
# See the Postfix UUCP_README file for configuration details.
#
uucp    unix  -       n       n       -       -       pipe
  flags=Fqhu user=uucp argv=uux -r -n -z -a$sender - $nexthop!rmail ($recipient)
#
# Other external delivery methods.
#
ifmail   unix  -       n       n       -       -       pipe
  flags=F user=ftn argv=/usr/lib/ifmail/ifmail -r $nexthop ($recipient)
bsmtp  unix  -       n       n       -       -       pipe
  flags=Fq. user=bsmtp argv=/usr/lib/bsmtp/bsmtp -t$nexthop -f$sender $recipient
scalemail-backend unix   -   n   n   -   2   pipe
  flags=R user=scalemail argv=/usr/lib/scalemail/bin/scalemail-store ${nexthop} ${user} ${extension}
mailman   unix  -     n       n       -       -       pipe
  flags=FR user=list argv=/usr/lib/mailman/bin/postfix-to-mailman.py
  ${nexthop} ${user}
```

/etc/opendkim.conf

```
# This is a basic configuration that can easily be adapted to suit a standard
# installation. For more advanced options, see opendkim.conf(5) and/or
# /usr/share/doc/opendkim/examples/opendkim.conf.sample.

# Log to syslog
Syslog                  yes
# Required to use local socket with MTAs that access the socket as a non-
# privileged user (e.g. Postfix)
UMask                   007

# Sign for example.com with key in /etc/dkimkeys/dkim.key using
# selector '2007' (e.g. 2007._domainkey.example.com)
#Domain                 example.com
#KeyFile         /etc/dkimkeys/dkim.key
#Selector        2007

# Commonly-used options; the commented-out versions show the defaults.
Canonicalization    relaxed/simple
Mode                    s
SubDomains              no

# Socket smtp://localhost
#
# ##

#Socket socketspec
# ##
# ##  Names the socket where this filter should listen for milter connections
# ##  from the MTA.  Required.  Should be in one of these forms:
# ##
# ##  inet:port@address        to listen on a specific interface
# ##  inet:port                to listen on all interfaces
# ##  local:/path/to/socket    to listen on a UNIX domain socket
#
#Socket                 inet:8892@localhost
Socket                  local:/var/spool/postfix/opendkim/opendkim.sock

##  PidFile filename
###     default (none)
###
###  Name of the file where the filter should write its pid before beginning
###  normal operations.
#
PidFile          /var/run/opendkim/opendkim.pid


# Always oversign From (sign using actual From and a null From to prevent
# malicious signatures header fields (From and/or others) between the signer
# and the verifier.  From is oversigned by default in the Debian pacakge
# because it is often the identity key used by reputation systems and thus
# somewhat security sensitive.
OversignHeaders        From

##  ResolverConfiguration filename
##       default (none)
##
##  Specifies a configuration file to be passed to the Unbound library that
##  performs DNS queries applying the DNSSEC protocol.  See the Unbound
##  documentation at http://unbound.net for the expected content of this file.
##  The results of using this and the TrustAnchorFile setting at the same
##  time are undefined.
##  In Debian, /etc/unbound/unbound.conf is shipped as part of the Suggested
##  unbound package

# ResolverConfiguration        /etc/unbound/unbound.conf

##  TrustAnchorFile filename
##       default (none)
##
## Specifies a file from which trust anchor data should be read when doing
## DNS queries and applying the DNSSEC protocol.  See the Unbound documentation
## at http://unbound.net for the expected format of this file.

TrustAnchorFile         /usr/share/dns/root.key

##  Userid userid
###      default (none)
###
###  Change to user "userid" before starting normal operation?  May include
###  a group ID as well, separated from the userid by a colon.
#
UserID            opendkim

# Map domains in From addresses to keys used to sign messages

KeyTable refile:/etc/opendkim/key.table

SigningTable refile:/etc/opendkim/signing.table

# A set of internal hosts whose mail should be signed

InternalHosts /etc/opendkim/trusted.hosts
#RequireSafeKeys false
```

/etc/socket/opendkim

```
# Command-line options specified here will override the contents of
# /etc/opendkim.conf. See opendkim(8) for a complete list of options.
#DAEMON_OPTS=""
# Change to /var/spool/postfix/var/run/opendkim to use a Unix socket with
# postfix in a chroot:
#RUNDIR=/var/spool/postfix/var/run/opendkim
RUNDIR=/var/run/opendkim
#
# Uncomment to specify an alternate socket
# Note that setting this will override any Socket value in opendkim.conf
# default:
SOCKET="local:/var/spool/postfix/opendkim/opendkim.sock"
# listen on all interfaces on port 54321:
#SOCKET=inet:54321
# listen on loopback on port 12345:
#SOCKET=inet:12345@localhost
# listen on 192.0.2.1 on port 12345:
#SOCKET=inet:12345@192.0.2.1
USER=opendkim
GROUP=opendkim
PIDFILE=$RUNDIR/$NAME.pid
EXTRAAFTER=
```

/etc/dovecot/conf.d/10-auth.conf

```
##
## Authentication processes
##

# Disable LOGIN command and all other plaintext authentications unless
# SSL/TLS is used (LOGINDISABLED capability). Note that if the remote IP
# matches the local IP (ie. you're connecting from the same computer), the
# connection is considered secure and plaintext authentication is allowed.
# See also ssl=required setting.
disable_plaintext_auth = yes

# Authentication cache size (e.g. 10M). 0 means it's disabled. Note that
# bsdauth, PAM and vpopmail require cache_key to be set for caching to be used.
#auth_cache_size = 0
# Time to live for cached data. After TTL expires the cached record is no
# longer used, *except* if the main database lookup returns internal failure.
# We also try to handle password changes automatically: If user's previous
# authentication was successful, but this one wasn't, the cache isn't used.
# For now this works only with plaintext authentication.
#auth_cache_ttl = 1 hour
# TTL for negative hits (user not found, password mismatch).
# 0 disables caching them completely.
#auth_cache_negative_ttl = 1 hour

# Space separated list of realms for SASL authentication mechanisms that need
# them. You can leave it empty if you don't want to support multiple realms.
# Many clients simply use the first one listed here, so keep the default realm
# first.
#auth_realms =

# Default realm/domain to use if none was specified. This is used for both
# SASL realms and appending @domain to username in plaintext logins.
#auth_default_realm =

# List of allowed characters in username. If the user-given username contains
# a character not listed in here, the login automatically fails. This is just
# an extra check to make sure user can't exploit any potential quote escaping
# vulnerabilities with SQL/LDAP databases. If you want to allow all characters,
# set this value to empty.
#auth_username_chars =
abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ01234567890.-_@

# Username character translations before it's looked up from databases. The
# value contains series of from -> to characters. For example "#@/@" means
# that '#' and '/' characters are translated to '@'.
#auth_username_translation =

# Username formatting before it's looked up from databases. You can use
# the standard variables here, eg. %Lu would lowercase the username, %n would
# drop away the domain if it was given, or "%n-AT-%d" would change the '@' into
# "-AT-". This translation is done after auth_username_translation changes.
#auth_username_format = %Lu

# If you want to allow master users to log in by specifying the master
# username within the normal username string (ie. not using SASL mechanism's
# support for it), you can specify the separator character here. The format
# is then <username><separator><master username>. UW-IMAP uses "*" as the
# separator, so that could be a good choice.
#auth_master_user_separator =

# Username to use for users logging in with ANONYMOUS SASL mechanism
#auth_anonymous_username = anonymous

# Maximum number of dovecot-auth worker processes. They're used to execute
# blocking passdb and userdb queries (eg. MySQL and PAM). They're
# automatically created and destroyed as needed.
#auth_worker_max_count = 30

# Host name to use in GSSAPI principal names. The default is to use the
# name returned by gethostname(). Use "$ALL" (with quotes) to allow all keytab
# entries.
#auth_gssapi_hostname =

# Kerberos keytab to use for the GSSAPI mechanism. Will use the system
# default (usually /etc/krb5.keytab) if not specified. You may need to change
# the auth service to run as root to be able to read this file.
#auth_krb5_keytab =

# Do NTLM and GSS-SPNEGO authentication using Samba's winbind daemon and
# ntlm_auth helper. <doc/wiki/Authentication/Mechanisms/Winbind.txt>
#auth_use_winbind = no

# Path for Samba's ntlm_auth helper binary.
#auth_winbind_helper_path = /usr/bin/ntlm_auth

# Time to delay before replying to failed authentications.
#auth_failure_delay = 2 secs

# Require a valid SSL client certificate or the authentication fails.
#auth_ssl_require_client_cert = no

# Take the username from client's SSL certificate, using
# X509_NAME_get_text_by_NID() which returns the subject's DN's
# CommonName.
#auth_ssl_username_from_cert = no

# Space separated list of wanted authentication mechanisms:
#   plain login digest-md5 cram-md5 ntlm rpa apop anonymous gssapi otp skey
#   gss-spnego
# NOTE: See also disable_plaintext_auth setting.
auth_mechanisms = plain login

##
## Password and user databases
##

#
# Password database is used to verify user's password (and nothing more).
# You can have multiple passdbs and userdbs. This is useful if you want to
# allow both system users (/etc/passwd) and virtual users to login without
# duplicating the system users into virtual database.
#
```

```
# <doc/wiki/PasswordDatabase.txt>
#
# User database specifies where mails are located and what user/group IDs
# own them. For single-UID configuration use "static" userdb.
#
# <doc/wiki/UserDatabase.txt>

#!include auth-deny.conf.ext
#!include auth-master.conf.ext

!include auth-system.conf.ext
#!include auth-sql.conf.ext
#!include auth-ldap.conf.ext
#!include auth-passwdfile.conf.ext
#!include auth-checkpassword.conf.ext
#!include auth-vpopmail.conf.ext
#!include auth-static.conf.ext
```

/etc/dovecot/conf.d/10-mail.conf

```
##
## Mailbox locations and namespaces
##

# Location for users' mailboxes. The default is empty, which means that Dovecot
# tries to find the mailboxes automatically. This won't work if the user
# doesn't yet have any mail, so you should explicitly tell Dovecot the full
# location.
#
# If you're using mbox, giving a path to the INBOX file (eg. /var/mail/%u)
# isn't enough. You'll also need to tell Dovecot where the other mailboxes are
# kept. This is called the "root mail directory", and it must be the first
# path given in the mail_location setting.
#
# There are a few special variables you can use, eg.:
#
#   %u - username
#   %n - user part in user@domain, same as %u if there's no domain
#   %d - domain part in user@domain, empty if there's no domain
#   %h - home directory
#
# 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

# If you need to set multiple mailbox locations or want to change default
# namespace settings, you can do it by defining namespace sections.
#
# You can have private, shared and public namespaces. Private namespaces
# are for user's personal mails. Shared namespaces are for accessing other
# users' mailboxes that have been shared. Public namespaces are for shared
# mailboxes that are managed by sysadmin. If you create any shared or public
# namespaces you'll typically want to enable ACL plugin also, otherwise all
# users can access all the shared mailboxes, assuming they have permissions
# on filesystem level to do so.
namespace inbox {
  # Namespace type: private, shared or public
  #type = private

  # Hierarchy separator to use. You should use the same separator for all
  # namespaces or some clients get confused. '/' is usually a good one.
  # The default however depends on the underlying mail storage format.
  #separator =

  # Prefix required to access this namespace. This needs to be different for
  # all namespaces. For example "Public/".
  #prefix =

  # Physical location of the mailbox. This is in same format as
  # mail_location, which is also the default for it.
  #location =

  # There can be only one INBOX, and this setting defines which namespace
  # has it.
  inbox = yes

  # If namespace is hidden, it's not advertised to clients via NAMESPACE
  # extension. You'll most likely also want to set list=no. This is mostly
  # useful when converting from another server with different namespaces which
  # you want to deprecate but still keep working. For example you can create
  # hidden namespaces with prefixes "~/mail/", "~%u/mail/" and "mail/".
  #hidden = no

  # Show the mailboxes under this namespace with LIST command. This makes the
  # namespace visible for clients that don't support NAMESPACE extension.
  # "children" value lists child mailboxes, but hides the namespace prefix.
  #list = yes

  # Namespace handles its own subscriptions. If set to "no", the parent
  # namespace handles them (empty prefix should always have this as "yes")
  #subscriptions = yes

  # See 15-mailboxes.conf for definitions of special mailboxes.
}

# Example shared namespace configuration
#namespace {
  #type = shared
  #separator = /

  # Mailboxes are visible under "shared/user@domain/"
  # %%n, %%d and %%u are expanded to the destination user.
  #prefix = shared/%%u/

  # Mail location for other users' mailboxes. Note that %variables and ~/
  # expands to the logged in user's data. %%n, %%d, %%u and %%h expand to the
  # destination user's data.
  #location = maildir:%%h/Maildir:INDEX=~/Maildir/shared/%%u

  # Use the default namespace for saving subscriptions.
  #subscriptions = no

  # List the shared/ namespace only if there are visible shared mailboxes.
  #list = children
#}
# Should shared INBOX be visible as "shared/user" or "shared/user/INBOX"?
#mail_shared_explicit_inbox = no

# System user and group used to access mails. If you use multiple, userdb
# can override these by returning uid or gid fields. You can use either numbers
# or names. <doc/wiki/UserIds.txt>
#mail_uid =
#mail_gid =

# Group to enable temporarily for privileged operations. Currently this is
# used only with INBOX when either its initial creation or dotlocking fails.
# Typically this is set to "mail" to give access to /var/mail.
#mail_privileged_group =

# Grant access to these supplementary groups for mail processes. Typically
# these are used to set up access to shared mailboxes. Note that it may be
# dangerous to set these if users can create symlinks (e.g. if "mail" group is
# set here, ln -s /var/mail ~/mail/var could allow a user to delete others'
# mailboxes, or ln -s /secret/shared/box ~/mail/mybox would allow reading it).
#mail_access_groups =

# Allow full filesystem access to clients. There's no access checks other than
# what the operating system does for the active UID/GID. It works with both
# maildir and mboxes, allowing you to prefix mailboxes names with eg. /path/
# or ~user/.
#mail_full_filesystem_access = no

# Dictionary for key=value mailbox attributes. This is used for example by
# URLAUTH and METADATA extensions.
#mail_attribute_dict =

# A comment or note that is associated with the server. This value is
# accessible for authenticated users through the IMAP METADATA server
# entry "/shared/comment".
#mail_server_comment = ""

# Indicates a method for contacting the server administrator. According to
# RFC 5464, this value MUST be a URI (e.g., a mailto: or tel: URL), but that
# is currently not enforced. Use for example mailto:admin@example.com. This
# value is accessible for authenticated users through the IMAP METADATA server
# entry "/shared/admin".
#mail_server_admin =

##
## Mail processes
##

# Don't use mmap() at all. This is required if you store indexes to shared
# filesystems (NFS or clustered filesystem).
#mmap_disable = no

# Rely on O_EXCL to work when creating dotlock files. NFS supports O_EXCL
# since version 3, so this should be safe to use nowadays by default.
#dotlock_use_excl = yes

# When to use fsync() or fdatasync() calls:
#   optimized (default): Whenever necessary to avoid losing important data
#   always: Useful with e.g. NFS when write()s are delayed
#   never: Never use it (best performance, but crashes can lose data)
#mail_fsync = optimized

# Locking method for index files. Alternatives are fcntl, flock and dotlock.
# Dotlocking uses some tricks which may create more disk I/O than other locking
# methods. NFS users: flock doesn't work, remember to change mmap_disable.
#lock_method = fcntl

# Directory in which LDA/LMTP temporarily stores incoming mails >128 kB.
#mail_temp_dir = /tmp

# Valid UID range for users, defaults to 500 and above. This is mostly
# to make sure that users can't log in as daemons or other system users.
# Note that denying root logins is hardcoded to dovecot binary and can't
# be done even if first_valid_uid is set to 0.
#first_valid_uid = 500
#last_valid_uid = 0

# Valid GID range for users, defaults to non-root/wheel. Users having
# non-valid GID as primary group ID aren't allowed to log in. If user
# belongs to supplementary groups with non-valid GIDs, those groups are
# not set.
#first_valid_gid = 1
#last_valid_gid = 0

# Maximum allowed length for mail keyword name. It's only forced when trying
# to create new keywords.
#mail_max_keyword_length = 50

# ':' separated list of directories under which chrooting is allowed for mail
# processes (ie. /var/mail will allow chrooting to /var/mail/foo/bar too).
# This setting doesn't affect login_chroot, mail_chroot or auth chroot
# settings. If this setting is empty, "/./" in home dirs are ignored.
# WARNING: Never add directories here which local users can modify, that
# may lead to root exploit. Usually this should be done only if you don't
# allow shell access for users. <doc/wiki/Chrooting.txt>
#valid_chroot_dirs =

# Default chroot directory for mail processes. This can be overridden for
# specific users in user database by giving /./ in user's home directory
# (eg. /home/./user chroots into /home). Note that usually there is no real
# need to do chrooting, Dovecot doesn't allow users to access files outside
# their mail directory anyway. If your home directories are prefixed with
# the chroot directory, append "/." to mail_chroot. <doc/wiki/Chrooting.txt>
#mail_chroot =

# UNIX socket path to master authentication server to find users.
# This is used by imap (for shared users) and lda.
#auth_socket_path = /var/run/dovecot/auth-userdb

# Directory where to look up mail plugins.
#mail_plugin_dir = /usr/lib/dovecot/modules

# Space separated list of plugins to load for all services. Plugins specific to
# IMAP, LDA, etc. are added to this list in their own .conf files.
#mail_plugins =

##
## Mailbox handling optimizations
##

# Mailbox list indexes can be used to optimize IMAP STATUS commands. They are
# also required for IMAP NOTIFY extension to be enabled.
#mailbox_list_index = no

# The minimum number of mails in a mailbox before updates are done to cache
# file. This allows optimizing Dovecot's behavior to do less disk writes at
# the cost of more disk reads.
#mail_cache_min_mail_count = 0

# When IDLE command is running, mailbox is checked once in a while to see if
# there are any new mails or other changes. This setting defines the minimum
# time to wait between those checks. Dovecot can also use inotify and
# kqueue to find out immediately when changes occur.
#mailbox_idle_check_interval = 30 secs

# Save mails with CR+LF instead of plain LF. This makes sending those mails
# take less CPU, especially with sendfile() syscall with Linux and FreeBSD.
# But it also creates a bit more disk I/O which may just make it slower.
# Also note that if other software reads the mboxes/maildirs, they may handle
# the extra CRs wrong and cause problems.
#mail_save_crlf = no

# Max number of mails to keep open and prefetch to memory. This only works with
# some mailbox formats and/or operating systems.
#mail_prefetch_count = 0

# How often to scan for stale temporary files and delete them (0 = never).
# These should exist only after Dovecot dies in the middle of saving mails.
#mail_temp_scan_interval = 1w

##
## Maildir-specific settings
##
```

```
# By default LIST command returns all entries in maildir beginning with a dot.
# Enabling this option makes Dovecot return only entries which are directories.
# This is done by stat()ing each entry, so it causes more disk I/O.
# (For systems setting struct dirent->d_type, this check is free and it's
# done always regardless of this setting)
#maildir_stat_dirs = no

# When copying a message, do it with hard links whenever possible. This makes
# the performance much better, and it's unlikely to have any side effects.
#maildir_copy_with_hardlinks = yes

# Assume Dovecot is the only MUA accessing Maildir: Scan cur/ directory only
# when its mtime changes unexpectedly or when we can't find the mail otherwise.
#maildir_very_dirty_syncs = no

# If enabled, Dovecot doesn't use the S=<size> in the Maildir filenames for
# getting the mail's physical size, except when recalculating Maildir++ quota.
# This can be useful in systems where a lot of the Maildir filenames have a
# broken size. The performance hit for enabling this is very small.
#maildir_broken_filename_sizes = no

# Always move mails from new/ directory to cur/, even when the \Recent flags
# aren't being reset.
#maildir_empty_new = no

##
## mbox-specific settings
##

# Which locking methods to use for locking mbox. There are four available:
#  dotlock: Create <mailbox>.lock file. This is the oldest and most NFS-safe
#           solution. If you want to use /var/mail/ like directory, the users
#           will need write access to that directory.
#  dotlock_try: Same as dotlock, but if it fails because of permissions or
#               because there isn't enough disk space, just skip it.
#  fcntl  : Use this if possible. Works with NFS too if lockd is used.
#  flock  : May not exist in all systems. Doesn't work with NFS.
#  lockf  : May not exist in all systems. Doesn't work with NFS.
#
# You can use multiple locking methods; if you do the order they're declared
# in is important to avoid deadlocks if other MTAs/MUAs are using multiple
# locking methods as well. Some operating systems don't allow using some of
# them simultaneously.
#
# The Debian value for mbox_write_locks differs from upstream Dovecot. It is
# changed to be compliant with Debian Policy (section 11.6) for NFS safety.
#         Dovecot: mbox_write_locks = dotlock fcntl
#         Debian:  mbox_write_locks = fcntl dotlock
#
#mbox_read_locks = fcntl
#mbox_write_locks = fcntl dotlock

# Maximum time to wait for lock (all of them) before aborting.
#mbox_lock_timeout = 5 mins

# If dotlock exists but the mailbox isn't modified in any way, override the
# lock file after this much time.
#mbox_dotlock_change_timeout = 2 mins

# When mbox changes unexpectedly we have to fully read it to find out what
# changed. If the mbox is large this can take a long time. Since the change
# is usually just a newly appended mail, it'd be faster to simply read the
# new mails. If this setting is enabled, Dovecot does this but still safely
# fallbacks to re-reading the whole mbox file whenever something in mbox isn't
# how it's expected to be. The only real downside to this setting is that if
# some other MUA changes message flags, Dovecot doesn't notice it immediately.
# Note that a full sync is done with SELECT, EXAMINE, EXPUNGE and CHECK
# commands.
#mbox_dirty_syncs = yes

# Like mbox_dirty_syncs, but don't do full syncs even with SELECT, EXAMINE,
# EXPUNGE or CHECK commands. If this is set, mbox_dirty_syncs is ignored.
#mbox_very_dirty_syncs = no

# Delay writing mbox headers until doing a full write sync (EXPUNGE and CHECK
# commands and when closing the mailbox). This is especially useful for POP3
# where clients often delete all mails. The downside is that our changes
# aren't immediately visible to other MUAs.
#mbox_lazy_writes = yes

# If mbox size is smaller than this (e.g. 100k), don't write index files.
# If an index file already exists it's still read, just not updated.
#mbox_min_index_size = 0

# Mail header selection algorithm to use for MD5 POP3 UIDLs when
# pop3_uidl_format=%m. For backwards compatibility we use apop3d inspired
# algorithm, but it fails if the first Received: header isn't unique in all
# mails. An alternative algorithm is "all" that selects all headers.
#mbox_md5 = apop3d

##
## mdbox-specific settings
##
```

```
# Maximum dbox file size until it's rotated.
#mdbox_rotate_size = 2M

# Maximum dbox file age until it's rotated. Typically in days. Day begins
# from midnight, so 1d = today, 2d = yesterday, etc. 0 = check disabled.
#mdbox_rotate_interval = 0

# When creating new mdbox files, immediately preallocate their size to
# mdbox_rotate_size. This setting currently works only in Linux with some
# filesystems (ext4, xfs).
#mdbox_preallocate_space = no

##
## Mail attachments
##

# sdbox and mdbox support saving mail attachments to external files, which
# also allows single instance storage for them. Other backends don't support
# this for now.

# Directory root where to store mail attachments. Disabled, if empty.
#mail_attachment_dir =

# Attachments smaller than this aren't saved externally. It's also possible to
# write a plugin to disable saving specific attachments externally.
#mail_attachment_min_size = 128k

# Filesystem backend to use for saving attachments:
#  posix : No SiS done by Dovecot (but this might help FS's own deduplication)
#  sis posix : SiS with immediate byte-by-byte comparison during saving
#  sis-queue posix : SiS with delayed comparison and deduplication
#mail_attachment_fs = sis posix

# Hash format to use in attachment filenames. You can add any text and
# variables: %{md4}, %{md5}, %{sha1}, %{sha256}, %{sha512}, %{size}.
# Variables can be truncated, e.g. %{sha256:80} returns only first 80 bits
#mail_attachment_hash = %{sha1}
```

/etc/dovecot/conf.d/10-master.conf

```
#default_process_limit = 100
#default_client_limit = 1000
```

```
# Default VSZ (virtual memory size) limit for service processes. This is mainly
# intended to catch and kill processes that leak memory before they eat up
# everything.
#default_vsz_limit = 256M

# Login user is internally used by login processes. This is the most untrusted
# user in Dovecot system. It shouldn't have access to anything at all.
#default_login_user = dovenull

# Internal user is used by unprivileged processes. It should be separate from
# login user, so that login processes can't disturb other processes.
#default_internal_user = dovecot

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

  # Number of connections to handle before starting a new process. Typically
  # the only useful values are 0 (unlimited) or 1. 1 is more secure, but 0
  # is faster. <doc/wiki/LoginProcess.txt>
  #service_count = 1

  # Number of processes to always keep waiting for more connections.
  #process_min_avail = 0

  # If you set service_count=0, you probably need to grow this.
  #vsz_limit = $default_vsz_limit
}

service pop3-login {
  inet_listener pop3 {
        port = 110
  }
  inet_listener pop3s {
        port = 995
        ssl = yes
  }
}

service lmtp {
  unix_listener lmtp {
        #mode = 0666
  }

  # Create inet listener only if you can't use the above UNIX socket
  #inet_listener lmtp {
        # Avoid making LMTP visible for the entire internet
        #address =
        #port =
  #}
}

service imap {
  # Most of the memory goes to mmap()ing files. You may need to increase this
  # limit if you have huge mailboxes.
  #vsz_limit = $default_vsz_limit

  # Max. number of IMAP processes (connections)
  #process_limit = 1024
}

service pop3 {
  # Max. number of POP3 processes (connections)
  #process_limit = 1024
}

service auth {
  # auth_socket_path points to this userdb socket by default. It's typically
  # used by dovecot-lda, doveadm, possibly imap process, etc. Users that have
  # full permissions to this socket are able to get a list of all usernames and
  # get the results of everyone's userdb lookups.
  #
  # The default 0666 mode allows anyone to connect to the socket, but the
  # userdb lookups will succeed only if the userdb returns an "uid" field that
  # matches the caller process's UID. Also if caller's uid or gid matches the
  # socket's uid or gid the lookup succeeds. Anything else causes a failure.
  #
  # To give the caller full permissions to lookup all users, set the mode to
  # something else than 0666 and Dovecot lets the kernel enforce the
  # permissions (e.g. 0777 allows everyone full permissions).
  unix_listener auth-userdb {
        #mode = 0666
        #user = postfix
        #group = postfix
  }

  # Postfix smtp-auth
  unix_listener /var/spool/postfix/private/auth {
        mode = 0666
```

```
          user = postfix
          group = postfix
}

  # Auth process is run as this user.
  #user = $default_internal_user
}

service auth-worker {
  # Auth worker process is run as root by default, so that it can access
  # /etc/shadow. If this isn't necessary, the user should be changed to
  # $default_internal_user.
  #user = root
}

service dict {
  # If dict proxy is used, mail processes should have access to its socket.
  # For example: mode=0660, group=vmail and global mail_access_groups=vmail
  unix_listener dict {
          #mode = 0600
          #user =
          #group =
  }
}
```

/etc/dovecot/conf.d/10-ssl.conf

```
##
## SSL settings
##

# SSL/TLS support: yes, no, required. <doc/wiki/SSL.txt>
ssl = required

# PEM encoded X.509 SSL/TLS certificate and private key. They're opened before
# dropping root privileges, so keep the key file unreadable by anyone but
# root. Included doc/mkcert.sh can be used to easily generate self-signed
# certificate, just make sure to update the domains in dovecot-openssl.cnf
ssl_cert = </etc/letsencrypt/live/yodfat.com/fullchain.pem
ssl_key = </etc/letsencrypt/live/yodfat.com/privkey.pem

# If key file is password protected, give the password here. Alternatively
# give it when starting dovecot with -p parameter. Since this file is often
# world-readable, you may want to place this setting instead to a different
# root owned 0600 file by using ssl_key_password = <path.
```

```
#ssl_key_password =

# PEM encoded trusted certificate authority. Set this only if you intend to use
# ssl_verify_client_cert=yes. The file should contain the CA certificate(s)
# followed by the matching CRL(s). (e.g. ssl_ca = </etc/ssl/certs/ca.pem)
#ssl_ca =

# Require that CRL check succeeds for client certificates.
#ssl_require_crl = yes

# Directory and/or file for trusted SSL CA certificates. These are used only
# when Dovecot needs to act as an SSL client (e.g. imapc backend). The
# directory is usually /etc/ssl/certs in Debian-based systems and the file is
# /etc/pki/tls/cert.pem in RedHat-based systems.
#ssl_client_ca_dir =
#ssl_client_ca_file =

# Request client to send a certificate. If you also want to require it, set
# auth_ssl_require_client_cert=yes in auth section.
#ssl_verify_client_cert = no

# Which field from certificate to use for username. commonName and
# x500UniqueIdentifier are the usual choices. You'll also need to set
# auth_ssl_username_from_cert=yes.
#ssl_cert_username_field = commonName

# DH parameters length to use.
#ssl_dh_parameters_length = 1024

# SSL protocols to use
ssl_protocols = !SSLv3

# SSL ciphers to use
#ssl_cipher_list = ALL:!LOW:!SSLv2:!EXP:!aNULL

# Prefer the server's order of ciphers over client's.
#ssl_prefer_server_ciphers = no

# SSL crypto device to use, for valid values run "openssl engine"
#ssl_crypto_device =

# SSL extra options. Currently supported options are:
#   no_compression - Disable compression.
#ssl_options =
```