Red Hat Enterprise Linux 9

Deploying mail servers

Configuring and maintaining mail server services

Last Updated: 2024-05-01

Red Hat Enterprise Linux 9 Deploying mail servers

Configuring and maintaining mail server services

Legal Notice

Abstract

On Red Hat Enterprise Linux, you can provide reliable and secure mail services for your customers and internal users by using the mail transport agent Postfix as SMTP service and the mail delivery agent Dovecot as IMAP and POP3 services. Both services integrate with each other and they support central backends, such as LDAP directories to store account data and to authenticate users.

Table of Contents

MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright's message.

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your feedback on our documentation. Let us know how we can improve it.

Submitting feedback through Jira (account required)

1. Log in to the Jira website.
2. Click **Create** in the top navigation bar
3. Enter a descriptive title in the **Summary** field.
4. Enter your suggestion for improvement in the **Description** field. Include links to the relevant parts of the documentation.
5. Click **Create** at the bottom of the dialogue.

CHAPTER 1. CONFIGURING AND MAINTAINING A DOVECOT IMAP AND POP3 SERVER

Dovecot is a high-performance mail delivery agent (MDA) with a focus on security. You can use IMAP or POP3-compatible email clients to connect to a Dovecot server and read or download emails.

Key features of Dovecot:

- The design and implementation focuses on security
- Two-way replication support for high availability to improve the performance in large environments
- Supports the high-performance **dbox** mailbox format, but also **mbox** and **Maildir** for compatibility reasons
- Self-healing features, such as fixing broken index files
- Compliance with the IMAP standards
- Workaround support to bypass bugs in IMAP and POP3 clients

1.1. SETTING UP A DOVECOT SERVER WITH PAM AUTHENTICATION

Dovecot supports the Name Service Switch (NSS) interface as a user database and the Pluggable Authentication Modules (PAM) framework as an authentication backend. With this configuration, Dovecot can provide services to users who are available locally on the server through NSS.

Use PAM authentication if accounts:

- Are defined locally in the /**etc**/**passwd** file
- Are stored in a remote database but they are available locally through the System Security Services Daemon (SSSD) or other NSS plugins.

1.1.1. Installing Dovecot

The **dovecot** package provides:

- The **dovecot** service and the utilities to maintain it
- Services that Dovecot starts on demand, such as for authentication
- Plugins, such as server-side mail filtering
- Configuration files in the /**etc**/**dovecot**/ directory
- Documentation in the /**usr**/**share**/**doc**/**dovecot**/ directory

Procedure

- Install the **dovecot** package:

```
# dnf install dovecot
```

NOTE

If Dovecot is already installed and you require clean configuration files, rename or remove the **/etc/dovecot/** directory. Afterwards, reinstall the package. Without removing the configuration files, the **dnf reinstall dovecot** command does not reset the configuration files in **/etc/dovecot/**.

Next step

- Configuring TLS encryption on a Dovecot server .

1.1.2. Configuring TLS encryption on a Dovecot server

Dovecot provides a secure default configuration. For example, TLS is enabled by default to transmit credentials and data encrypted over networks. To configure TLS on a Dovecot server, you only need to set the paths to the certificate and private key files. Additionally, you can increase the security of TLS connections by generating and using Diffie-Hellman parameters to provide perfect forward secrecy (PFS).

Prerequisites

- Dovecot is installed.
- The following files have been copied to the listed locations on the server:
  - The server certificate: **/etc/pki/dovecot/certs/server.example.com.crt**
  - The private key: **/etc/pki/dovecot/private/server.example.com.key**
  - The Certificate Authority (CA) certificate: **/etc/pki/dovecot/certs/ca.crt**
- The hostname in the **Subject DN** field of the server certificate matches the server's Fully-qualified Domain Name (FQDN).
- If the server runs RHEL 9.2 or later and the FIPS mode is enabled, clients must either support the Extended Master Secret (EMS) extension or use TLS 1.3. TLS 1.2 connections without EMS fail. For more information, see the TLS extension "Extended Master Secret" enforced Knowledgebase article.

Procedure

1. Set secure permissions on the private key file:

```
# chown root:root /etc/pki/dovecot/private/server.example.com.key
# chmod 600 /etc/pki/dovecot/private/server.example.com.key
```

2. Generate a file with Diffie-Hellman parameters:

```
# openssl dhparam -out /etc/dovecot/dh.pem 4096
```

   Depending on the hardware and entropy on the server, generating Diffie-Hellman parameters with 4096 bits can take several minutes.

3. Set the paths to the certificate and private key files in the **/etc/dovecot/conf.d/10-ssl.conf** file:
   a. Update the **ssl_cert** and **ssl_key** parameters, and set them to use the paths of the server's

certificate and private key:

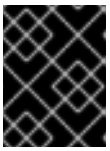
```
ssl_cert = </etc/pki/dovecot/certs/server.example.com.crt
ssl_key = </etc/pki/dovecot/private/server.example.com.key
```

b. Uncomment the **ssl_ca** parameter, and set it to use the path to the CA certificate:

```
ssl_ca = </etc/pki/dovecot/certs/ca.crt
```

c. Uncomment the **ssl_dh** parameter, and set it to use the path to the Diffie-Hellman parameters file:

```
ssl_dh = </etc/dovecot/dh.pem
```



IMPORTANT

To ensure that Dovecot reads the value of a parameter from a file, the path must start with a leading **<** character.

Next step

- Preparing Dovecot to use virtual users

Additional resources

- **/usr/share/doc/dovecot/wiki/SSL.DovecotConfiguration.txt**

1.1.3. Preparing Dovecot to use virtual users

By default, Dovecot performs many actions on the file system as the user who uses the service. However, configuring the Dovecot back end to use one local user to perform these actions has several benefits:

- Dovecot performs file system actions as a specific local user instead of using the user's ID (UID).
- Users do not need to be available locally on the server.
- You can store all mailboxes and user-specific files in one root directory.
- Users do not require a UID and group ID (GID), which reduces administration efforts.
- Users who have access to the file system on the server cannot compromise their mailboxes or indexes because they cannot access these files.
- Setting up replication is easier.

Prerequisites

- Dovecot is installed.

Procedure

1. Create the **vmail** user:

```
# useradd --home-dir /var/mail/ --shell /usr/sbin/nologin vmail
```

Dovecot will later use this user to manage the mailboxes. For security reasons, do not use the **dovecot** or **dovenull** system users for this purpose.

2. If you use a different path than **/var/mail/**, set the **mail_spool_t** SELinux context on it, for example:

```
# semanage fcontext -a -t mail_spool_t "<path>(/.*)?"
# restorecon -Rv <path>
```

3. Grant write permissions on **/var/mail/** only to the **vmail** user:

```
# chown vmail:vmail /var/mail/
# chmod 700 /var/mail/
```

4. Uncomment the **mail_location** parameter in the **/etc/dovecot/conf.d/10-mail.conf** file, and set it to the mailbox format and location:

```
mail_location = sdbox:/var/mail/%n/
```

With this setting:

- Dovecot uses the high-performant **dbox** mailbox format in **single** mode. In this mode, the service stores each mail in a separate file, similar to the **maildir** format.
- Dovecot resolves the **%n** variable in the path to the username. This is required to ensure that each user has a separate directory for its mailbox.

Next step

- Using PAM as the Dovecot authentication backend .

Additional resources

- **/usr/share/doc/dovecot/wiki/VirtualUsers.txt**
- **/usr/share/doc/dovecot/wiki/MailLocation.txt**
- **/usr/share/doc/dovecot/wiki/MailboxFormat.dbox.txt**
- **/usr/share/doc/dovecot/wiki/Variables.txt**

1.1.4. Using PAM as the Dovecot authentication backend

By default, Dovecot uses the Name Service Switch (NSS) interface as the user database and the Pluggable Authentication Modules (PAM) framework as the authentication backend.

Customize the settings to adapt Dovecot to your environment and to simplify administration by using the virtual users feature.

Prerequisites

- Dovecot is installed.

- The virtual users feature is configured.

Procedure

1. Update the **first_valid_uid** parameter in the **/etc/dovecot/conf.d/10-mail.conf** file to define the lowest user ID (UID) that can authenticate to Dovecot:

```
first_valid_uid = 1000
```

   By default, users with a UID greater than or equal to **1000** can authenticate. If required, you can also set the **last_valid_uid** parameter to define the highest UID that Dovecot allows to log in.

2. In the **/etc/dovecot/conf.d/auth-system.conf.ext** file, add the **override_fields** parameter to the **userdb** section as follows:

```
userdb {
  driver = passwd
  override_fields = uid=vmail gid=vmail home=/var/mail/%n/
}
```

   Due to the fixed values, Dovecot does not query these settings from the **/etc/passwd** file. As a result, the home directory defined in **/etc/passwd** does not need to exist.

Next step

- Complete the Dovecot configuration.

Additional resources

- **/usr/share/doc/dovecot/wiki/PasswordDatabase.PAM.txt**
- **/usr/share/doc/dovecot/wiki/VirtualUsers.Home.txt**

1.1.5. Completing the Dovecot configuration

Once you have installed and configured Dovecot, open the required ports in the **firewalld** service, and enable and start the service. Afterwards, you can test the server.

Prerequisites

- The following has been configured in Dovecot:
  - TLS encryption
  - An authentication backend
- Clients trust the Certificate Authority (CA) certificate.

Procedure

1. If you want to provide only an IMAP or POP3 service to users, uncomment the **protocols** parameter in the **/etc/dovecot/dovecot.conf** file, and set it to the required protocols. For example, if you do not require POP3, set:

```
protocols = imap lmtp
```

By default, the **imap**, **pop3**, and **lmtp** protocols are enabled.

2. Open the ports in the local firewall. For example, to open the ports for the IMAPS, IMAP, POP3S, and POP3 protocols, enter:

```
# firewall-cmd --permanent --add-service=imaps --add-service=imap --add-service=pop3s --add-service=pop3
# firewall-cmd --reload
```

3. Enable and start the **dovecot** service:

```
# systemctl enable --now dovecot
```

Verification

1. Use a mail client, such as Mozilla Thunderbird, to connect to Dovecot and read emails. The settings for the mail client depend on the protocol you want to use:

Table 1.1. Connection settings to the Dovecot server

Protocol	Port	Connection security	Authentication method
IMAP	143	STARTTLS	PLAIN[a]
IMAPS	993	SSL/TLS	PLAIN[a]
POP3	110	STARTTLS	PLAIN[a]
POP3S	995	SSL/TLS	PLAIN[a]

[a] The client transmits data encrypted through the TLS connection. Consequently, credentials are not disclosed.

Note that this table does not list settings for unencrypted connections because, by default, Dovecot does not accept plain text authentication on connections without TLS.

2. Display configuration settings with non-default values:

```
# doveconf -n
```

Additional resources

- **firewall-cmd(1)** man page

1.2. SETTING UP A DOVECOT SERVER WITH LDAP AUTHENTICATION

If your infrastructure uses an LDAP server to store accounts, you can authenticate Dovecot users against it. In this case, you manage accounts centrally in the directory and, users do not required local access to the file system on the Dovecot server.

Centrally-managed accounts are also a benefit if you plan to set up multiple Dovecot servers with replication to make your mailboxes high available.

1.2.1. Installing Dovecot

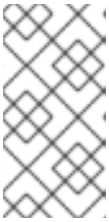
The **dovecot** package provides:

- The **dovecot** service and the utilities to maintain it
- Services that Dovecot starts on demand, such as for authentication
- Plugins, such as server-side mail filtering
- Configuration files in the **/etc/dovecot/** directory
- Documentation in the **/usr/share/doc/dovecot/** directory

Procedure

- Install the **dovecot** package:

```
# dnf install dovecot
```



NOTE

If Dovecot is already installed and you require clean configuration files, rename or remove the **/etc/dovecot/** directory. Afterwards, reinstall the package. Without removing the configuration files, the **dnf reinstall dovecot** command does not reset the configuration files in **/etc/dovecot/**.

Next step

- Configuring TLS encryption on a Dovecot server .

1.2.2. Configuring TLS encryption on a Dovecot server

Dovecot provides a secure default configuration. For example, TLS is enabled by default to transmit credentials and data encrypted over networks. To configure TLS on a Dovecot server, you only need to set the paths to the certificate and private key files. Additionally, you can increase the security of TLS connections by generating and using Diffie-Hellman parameters to provide perfect forward secrecy (PFS).

Prerequisites

- Dovecot is installed.
- The following files have been copied to the listed locations on the server:
  - The server certificate: **/etc/pki/dovecot/certs/server.example.com.crt**
  - The private key: **/etc/pki/dovecot/private/server.example.com.key**

- The Certificate Authority (CA) certificate: **/etc/pki/dovecot/certs/ca.crt**

- The hostname in the **Subject DN** field of the server certificate matches the server's Fully-qualified Domain Name (FQDN).

- If the server runs RHEL 9.2 or later and the FIPS mode is enabled, clients must either support the Extended Master Secret (EMS) extension or use TLS 1.3. TLS 1.2 connections without EMS fail. For more information, see the TLS extension "Extended Master Secret" enforced Knowledgebase article.

Procedure

1. Set secure permissions on the private key file:

```
# chown root:root /etc/pki/dovecot/private/server.example.com.key
# chmod 600 /etc/pki/dovecot/private/server.example.com.key
```

2. Generate a file with Diffie-Hellman parameters:

```
# openssl dhparam -out /etc/dovecot/dh.pem 4096
```

   Depending on the hardware and entropy on the server, generating Diffie-Hellman parameters with 4096 bits can take several minutes.

3. Set the paths to the certificate and private key files in the **/etc/dovecot/conf.d/10-ssl.conf** file:

   a. Update the **ssl_cert** and **ssl_key** parameters, and set them to use the paths of the server's certificate and private key:

```
ssl_cert = </etc/pki/dovecot/certs/server.example.com.crt
ssl_key = </etc/pki/dovecot/private/server.example.com.key
```

   b. Uncomment the **ssl_ca** parameter, and set it to use the path to the CA certificate:

```
ssl_ca = </etc/pki/dovecot/certs/ca.crt
```

   c. Uncomment the **ssl_dh** parameter, and set it to use the path to the Diffie-Hellman parameters file:

```
ssl_dh = </etc/dovecot/dh.pem
```

IMPORTANT

To ensure that Dovecot reads the value of a parameter from a file, the path must start with a leading **<** character.

Next step

- Preparing Dovecot to use virtual users

Additional resources

- **/usr/share/doc/dovecot/wiki/SSL.DovecotConfiguration.txt**

1.2.3. Preparing Dovecot to use virtual users

By default, Dovecot performs many actions on the file system as the user who uses the service. However, configuring the Dovecot back end to use one local user to perform these actions has several benefits:

- Dovecot performs file system actions as a specific local user instead of using the user's ID (UID).
- Users do not need to be available locally on the server.
- You can store all mailboxes and user-specific files in one root directory.
- Users do not require a UID and group ID (GID), which reduces administration efforts.
- Users who have access to the file system on the server cannot compromise their mailboxes or indexes because they cannot access these files.
- Setting up replication is easier.

Prerequisites

- Dovecot is installed.

Procedure

1. Create the **vmail** user:

```
# useradd --home-dir /var/mail/ --shell /usr/sbin/nologin vmail
```

   Dovecot will later use this user to manage the mailboxes. For security reasons, do not use the **dovecot** or **dovenull** system users for this purpose.

2. If you use a different path than **/var/mail/**, set the **mail_spool_t** SELinux context on it, for example:

```
# semanage fcontext -a -t mail_spool_t "<path>(/.*)?"
# restorecon -Rv <path>
```

3. Grant write permissions on **/var/mail/** only to the **vmail** user:

```
# chown vmail:vmail /var/mail/
# chmod 700 /var/mail/
```

4. Uncomment the **mail_location** parameter in the **/etc/dovecot/conf.d/10-mail.conf** file, and set it to the mailbox format and location:

```
mail_location = sdbox:/var/mail/%n/
```

   With this setting:

   - Dovecot uses the high-performant **dbox** mailbox format in **single** mode. In this mode, the service stores each mail in a separate file, similar to the **maildir** format.
   - Dovecot resolves the **%n** variable in the path to the username. This is required to ensure that each user has a separate directory for its mailbox.

Next step

- Using LDAP as the Dovecot authentication backend .

Additional resources

- **/usr/share/doc/dovecot/wiki/VirtualUsers.txt**
- **/usr/share/doc/dovecot/wiki/MailLocation.txt**
- **/usr/share/doc/dovecot/wiki/MailboxFormat.dbox.txt**
- **/usr/share/doc/dovecot/wiki/Variables.txt**

1.2.4. Using LDAP as the Dovecot authentication backend

Users in an LDAP directory can usually authenticate themselves to the directory service. Dovecot can use this to authenticate users when they log in to the IMAP and POP3 services. This authentication method has a number of benefits, such as:

- Administrators can manage users centrally in the directory.
- The LDAP accounts do not require any special attributes. They only need to be able to authenticate to the LDAP server. Consequently, this method is independent from the password storage scheme used on the LDAP server.
- Users do not need to be available locally on the server through the Name Service Switch (NSS) interface and the Pluggable Authentication Modules (PAM) framework.

Prerequisites

- Dovecot is installed.
- The virtual users feature is configured.
- Connections to the LDAP server support TLS encryption.
- RHEL on the Dovecot server trusts the Certificate Authority (CA) certificate of the LDAP server.
- If users are stored in different trees in the LDAP directory, a dedicated LDAP account for Dovecot exists to search the directory. This account requires permissions to search for Distinguished Names (DNs) of other users.
- If the MariaDB server runs RHEL 9.2 or later and the FIPS mode is enabled, this Dovecot server supports the Extended Master Secret (EMS) extension or uses TLS 1.3. TLS 1.2 connections without EMS fail. For more information, see the TLS extension "Extended Master Secret" enforced Knowledgebase article.

Procedure

1. Configure the authentication backends in the **/etc/dovecot/conf.d/10-auth.conf** file:
   a. Comment out **include** statements for **auth-*.conf.ext** authentication backend configuration files that you do not require, for example:

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

b. Enable LDAP authentication by uncommenting the following line:

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

2. Edit the **/etc/dovecot/conf.d/auth-ldap.conf.ext** file, and add the **override_fields** parameter as follows to the **userdb** section:

```
userdb {
  driver = ldap
  args = /etc/dovecot/dovecot-ldap.conf.ext
  override_fields = uid=vmail gid=vmail home=/var/mail/%n/
}
```

Due to the fixed values, Dovecot does not query these settings from the LDAP server. Consequently, these attributes also do not have to be present.

3. Create the **/etc/dovecot/dovecot-ldap.conf.ext** file with the following settings:

a. Depending on the LDAP structure, configure one of the following:

- If users are stored in different trees in the LDAP directory, configure dynamic DN lookups:

```
dn = cn=dovecot_LDAP,dc=example,dc=com
dnpass = password
pass_filter = (&(objectClass=posixAccount)(uid=%n))
```

Dovecot uses the specified DN, password, and filter to search the DN of the authenticating user in the directory. In this search, Dovecot replaces **%n** in the filter with the username. Note that the LDAP search must return only one result.

- If all users are stored under a specific entry, configure a DN template:

```
auth_bind_userdn = cn=%n,ou=People,dc=example,dc=com
```

b. Enable authentication binds to the LDAP server to verify Dovecot users:

```
auth_bind = yes
```

c. Set the URL to the LDAP server:

```
uris = ldaps://LDAP-srv.example.com
```

For security reasons, only use encrypted connections using LDAPS or the **STARTTLS** command over the LDAP protocol. For the latter, additionally add **tls = yes** to the settings.

For a working certificate validation, the hostname of the LDAP server must match the hostname used in its TLS certificate.

d. Enable the verification of the LDAP server's TLS certificate:

```
tls_require_cert = hard
```

e. Set the base DN to the DN where to start searching for users:

```
base = ou=People,dc=example,dc=com
```

f. Set the search scope:

```
scope = onelevel
```

Dovecot searches with the **onelevel** scope only in the specified base DN and with the **subtree** scope also in subtrees.

4. Set secure permissions on the **/etc/dovecot/dovecot-ldap.conf.ext** file:

```
# chown root:root /etc/dovecot/dovecot-ldap.conf.ext
# chmod 600 /etc/dovecot/dovecot-ldap.conf.ext
```

Next step

- Complete the Dovecot configuration.

Additional resources

- **/usr/share/doc/dovecot/example-config/dovecot-ldap.conf.ext**
- **/usr/share/doc/dovecot/wiki/UserDatabase.Static.txt**
- **/usr/share/doc/dovecot/wiki/AuthDatabase.LDAP.txt**
- **/usr/share/doc/dovecot/wiki/AuthDatabase.LDAP.AuthBinds.txt**
- **/usr/share/doc/dovecot/wiki/AuthDatabase.LDAP.PasswordLookups.txt**

1.2.5. Completing the Dovecot configuration

Once you have installed and configured Dovecot, open the required ports in the **firewalld** service, and enable and start the service. Afterwards, you can test the server.

Prerequisites

- The following has been configured in Dovecot:
  - TLS encryption
  - An authentication backend
- Clients trust the Certificate Authority (CA) certificate.

Procedure

1. If you want to provide only an IMAP or POP3 service to users, uncomment the **protocols** parameter in the **/etc/dovecot/dovecot.conf** file, and set it to the required protocols. For example, if you do not require POP3, set:

```
protocols = imap lmtp
```

By default, the **imap**, **pop3**, and **lmtp** protocols are enabled.

2. Open the ports in the local firewall. For example, to open the ports for the IMAPS, IMAP, POP3S, and POP3 protocols, enter:

```
# firewall-cmd --permanent --add-service=imaps --add-service=imap --add-service=pop3s --add-service=pop3
# firewall-cmd --reload
```

3. Enable and start the **dovecot** service:

```
# systemctl enable --now dovecot
```

Verification

1. Use a mail client, such as Mozilla Thunderbird, to connect to Dovecot and read emails. The settings for the mail client depend on the protocol you want to use:

Table 1.2. Connection settings to the Dovecot server

Protocol	Port	Connection security	Authentication method
IMAP	143	STARTTLS	PLAIN[a]
IMAPS	993	SSL/TLS	PLAIN[a]
POP3	110	STARTTLS	PLAIN[a]
POP3S	995	SSL/TLS	PLAIN[a]

[a] The client transmits data encrypted through the TLS connection. Consequently, credentials are not disclosed.

Note that this table does not list settings for unencrypted connections because, by default, Dovecot does not accept plain text authentication on connections without TLS.

2. Display configuration settings with non-default values:

```
# doveconf -n
```

Additional resources

- **firewall-cmd(1)** man page

1.3. SETTING UP A DOVECOT SERVER WITH MARIADB SQL AUTHENTICATION

If you store users and passwords in a MariaDB SQL server, you can configure Dovecot to use it as the user database and authentication backend. With this configuration, you manage accounts centrally in a database, and users have no local access to the file system on the Dovecot server.

Centrally managed accounts are also a benefit if you plan to set up multiple Dovecot servers with replication to make your mailboxes highly available.

1.3.1. Installing Dovecot

The **dovecot** package provides:

- The **dovecot** service and the utilities to maintain it
- Services that Dovecot starts on demand, such as for authentication
- Plugins, such as server-side mail filtering
- Configuration files in the **/etc/dovecot/** directory
- Documentation in the **/usr/share/doc/dovecot/** directory

Procedure

- Install the **dovecot** package:

```
# dnf install dovecot
```

NOTE

If Dovecot is already installed and you require clean configuration files, rename or remove the **/etc/dovecot/** directory. Afterwards, reinstall the package. Without removing the configuration files, the **dnf reinstall dovecot** command does not reset the configuration files in **/etc/dovecot/**.

Next step

- Configuring TLS encryption on a Dovecot server .

1.3.2. Configuring TLS encryption on a Dovecot server

Dovecot provides a secure default configuration. For example, TLS is enabled by default to transmit credentials and data encrypted over networks. To configure TLS on a Dovecot server, you only need to set the paths to the certificate and private key files. Additionally, you can increase the security of TLS connections by generating and using Diffie-Hellman parameters to provide perfect forward secrecy (PFS).

Prerequisites

- Dovecot is installed.
- The following files have been copied to the listed locations on the server:
  - The server certificate: **/etc/pki/dovecot/certs/server.example.com.crt**
  - The private key: **/etc/pki/dovecot/private/server.example.com.key**
  - The Certificate Authority (CA) certificate: **/etc/pki/dovecot/certs/ca.crt**

- The hostname in the **Subject DN** field of the server certificate matches the server's Fully-qualified Domain Name (FQDN).
- If the server runs RHEL 9.2 or later and the FIPS mode is enabled, clients must either support the Extended Master Secret (EMS) extension or use TLS 1.3. TLS 1.2 connections without EMS fail. For more information, see the TLS extension "Extended Master Secret" enforced Knowledgebase article.

Procedure

1. Set secure permissions on the private key file:

```
# chown root:root /etc/pki/dovecot/private/server.example.com.key
# chmod 600 /etc/pki/dovecot/private/server.example.com.key
```

2. Generate a file with Diffie-Hellman parameters:

```
# openssl dhparam -out /etc/dovecot/dh.pem 4096
```

   Depending on the hardware and entropy on the server, generating Diffie-Hellman parameters with 4096 bits can take several minutes.

3. Set the paths to the certificate and private key files in the **/etc/dovecot/conf.d/10-ssl.conf** file:

   a. Update the **ssl_cert** and **ssl_key** parameters, and set them to use the paths of the server's certificate and private key:

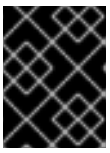
```
ssl_cert = </etc/pki/dovecot/certs/server.example.com.crt
ssl_key = </etc/pki/dovecot/private/server.example.com.key
```

   b. Uncomment the **ssl_ca** parameter, and set it to use the path to the CA certificate:

```
ssl_ca = </etc/pki/dovecot/certs/ca.crt
```

   c. Uncomment the **ssl_dh** parameter, and set it to use the path to the Diffie-Hellman parameters file:

```
ssl_dh = </etc/dovecot/dh.pem
```



IMPORTANT

To ensure that Dovecot reads the value of a parameter from a file, the path must start with a leading **<** character.

Next step

- Preparing Dovecot to use virtual users

Additional resources

- **/usr/share/doc/dovecot/wiki/SSL.DovecotConfiguration.txt**

1.3.3. Preparing Dovecot to use virtual users

By default, Dovecot performs many actions on the file system as the user who uses the service. However, configuring the Dovecot back end to use one local user to perform these actions has several benefits:

- Dovecot performs file system actions as a specific local user instead of using the user's ID (UID).
- Users do not need to be available locally on the server.
- You can store all mailboxes and user-specific files in one root directory.
- Users do not require a UID and group ID (GID), which reduces administration efforts.
- Users who have access to the file system on the server cannot compromise their mailboxes or indexes because they cannot access these files.
- Setting up replication is easier.

Prerequisites

- Dovecot is installed.

Procedure

1. Create the **vmail** user:

```
# useradd --home-dir /var/mail/ --shell /usr/sbin/nologin vmail
```

   Dovecot will later use this user to manage the mailboxes. For security reasons, do not use the **dovecot** or **dovenull** system users for this purpose.

2. If you use a different path than **/var/mail/**, set the **mail_spool_t** SELinux context on it, for example:

```
# semanage fcontext -a -t mail_spool_t "<path>(/.*)?"
# restorecon -Rv <path>
```

3. Grant write permissions on **/var/mail/** only to the **vmail** user:

```
# chown vmail:vmail /var/mail/
# chmod 700 /var/mail/
```

4. Uncomment the **mail_location** parameter in the **/etc/dovecot/conf.d/10-mail.conf** file, and set it to the mailbox format and location:

```
mail_location = sdbox:/var/mail/%n/
```

   With this setting:

   - Dovecot uses the high-performant **dbox** mailbox format in **single** mode. In this mode, the service stores each mail in a separate file, similar to the **maildir** format.
   - Dovecot resolves the **%n** variable in the path to the username. This is required to ensure that each user has a separate directory for its mailbox.

Next step

- Using a MariaDB SQL database as the Dovecot authentication backend

Additional resources

- **/usr/share/doc/dovecot/wiki/VirtualUsers.txt**
- **/usr/share/doc/dovecot/wiki/MailLocation.txt**
- **/usr/share/doc/dovecot/wiki/MailboxFormat.dbox.txt**
- **/usr/share/doc/dovecot/wiki/Variables.txt**

1.3.4. Using a MariaDB SQL database as the Dovecot authentication backend

Dovecot can read accounts and passwords from a MariaDB database and use it to authenticate users when they log in to the IMAP or POP3 service. The benefits of this authentication method include:

- Administrators can manage users centrally in a database.
- Users have no access locally on the server.

Prerequisites

- Dovecot is installed.
- The virtual users feature is configured.
- Connections to the MariaDB server support TLS encryption.
- The **dovecotDB** database exists in MariaDB, and the **users** table contains at least a **username** and **password** column.
- The **password** column contains passwords encrypted with a scheme that Dovecot supports.
- The passwords either use the same scheme or have a ***{pw-storage-scheme}*** prefix.
- The **dovecot** MariaDB user has read permission on the **users** table in the **dovecotDB** database.
- The certificate of the Certificate Authority (CA) that issued the MariaDB server's TLS certificate is stored on the Dovecot server in the **/etc/pki/tls/certs/ca.crt** file.
- If the MariaDB server runs RHEL 9.2 or later and the FIPS mode is enabled, this Dovecot server supports the Extended Master Secret (EMS) extension or uses TLS 1.3. TLS 1.2 connections without EMS fail. For more information, see the TLS extension "Extended Master Secret" enforced Knowledgebase article.

Procedure

1. Install the **dovecot-mysql** package:

```
# dnf install dovecot-mysql
```

2. Configure the authentication backends in the **/etc/dovecot/conf.d/10-auth.conf** file:

   a. Comment out **include** statements for **auth-*.conf.ext** authentication backend configuration files that you do not require, for example:

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

b. Enable SQL authentication by uncommenting the following line:

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

3. Edit the **/etc/dovecot/conf.d/auth-sql.conf.ext** file, and add the **override_fields** parameter to the **userdb** section as follows:

```
userdb {
  driver = sql
  args = /etc/dovecot/dovecot-sql.conf.ext
  override_fields = uid=vmail gid=vmail home=/var/mail/%n/
}
```

Due to the fixed values, Dovecot does not query these settings from the SQL server.

4. Create the **/etc/dovecot/dovecot-sql.conf.ext** file with the following settings:

```
driver = mysql
connect = host=mariadb_srv.example.com dbname=dovecotDB user=dovecot
password=dovecotPW ssl_ca=/etc/pki/tls/certs/ca.crt
default_pass_scheme = SHA512-CRYPT
user_query = SELECT username FROM users WHERE username='%u';
password_query = SELECT username AS user, password FROM users WHERE
username='%u';
iterate_query = SELECT username FROM users;
```

To use TLS encryption to the database server, set the **ssl_ca** option to the path of the certificate of the CA that issued the MariaDB server certificate. For a working certificate validation, the hostname of the MariaDB server must match the hostname used in its TLS certificate.

If the password values in the database contain a **{*pw-storage-scheme*}** prefix, you can omit the **default_pass_scheme** setting.

The queries in the file must be set as follows:

- For the **user_query** parameter, the query must return the username of the Dovecot user. The query must also return only one result.
- For the **password_query** parameter, the query must return the username and the password, and Dovecot must use these values in the **user** and **password** variables. Therefore, if the database uses different column names, use the **AS** SQL command to rename a column in the result.
- For the **iterate_query** parameter, the query must return a list of all users.

5. Set secure permissions on the **/etc/dovecot/dovecot-sql.conf.ext** file:

```
# chown root:root /etc/dovecot/dovecot-sql.conf.ext
# chmod 600 /etc/dovecot/dovecot-sql.conf.ext
```

Next step

- Complete the Dovecot configuration.

Additional resources

- **/usr/share/doc/dovecot/example-config/dovecot-sql.conf.ext**
- **/usr/share/doc/dovecot/wiki/Authentication.PasswordSchemes.txt**

1.3.5. Completing the Dovecot configuration

Once you have installed and configured Dovecot, open the required ports in the **firewalld** service, and enable and start the service. Afterwards, you can test the server.

Prerequisites

- The following has been configured in Dovecot:
  - TLS encryption
  - An authentication backend
- Clients trust the Certificate Authority (CA) certificate.

Procedure

1. If you want to provide only an IMAP or POP3 service to users, uncomment the **protocols** parameter in the **/etc/dovecot/dovecot.conf** file, and set it to the required protocols. For example, if you do not require POP3, set:

   ```
   protocols = imap lmtp
   ```

   By default, the **imap**, **pop3**, and **lmtp** protocols are enabled.

2. Open the ports in the local firewall. For example, to open the ports for the IMAPS, IMAP, POP3S, and POP3 protocols, enter:

   ```
   # firewall-cmd --permanent --add-service=imaps --add-service=imap --add-service=pop3s --add-service=pop3
   # firewall-cmd --reload
   ```

3. Enable and start the **dovecot** service:

   ```
   # systemctl enable --now dovecot
   ```

Verification

1. Use a mail client, such as Mozilla Thunderbird, to connect to Dovecot and read emails. The settings for the mail client depend on the protocol you want to use:

   Table 1.3. Connection settings to the Dovecot server

Protocol	Port	Connection security	Authentication method
IMAP	143	STARTTLS	PLAIN[a]
IMAPS	993	SSL/TLS	PLAIN[a]
POP3	110	STARTTLS	PLAIN[a]
POP3S	995	SSL/TLS	PLAIN[a]

[a] The client transmits data encrypted through the TLS connection. Consequently, credentials are not disclosed.

Note that this table does not list settings for unencrypted connections because, by default, Dovecot does not accept plain text authentication on connections without TLS.

2. Display configuration settings with non-default values:

```
# doveconf -n
```

Additional resources

- **firewall-cmd(1)** man page

1.4. CONFIGURING REPLICATION BETWEEN TWO DOVECOT SERVERS

With two-way replication, you can make your Dovecot server high-available, and IMAP and POP3 clients can access a mailbox on both servers. Dovecot keeps track of changes in the index logs of each mailbox and solves conflicts in a safe way.

Perform this procedure on both replication partners.

NOTE

Replication works only between server pairs. Consequently, in a large cluster, you need multiple independent backend pairs.

Prerequisites

- Both servers use the same authentication backend. Preferably, use LDAP or SQL to maintain accounts centrally.
- The Dovecot user database configuration supports user listing. Use the **doveadm user '*'** command to verify this.
- Dovecot accesses mailboxes on the file system as the **vmail** user instead of the user's ID (UID).

Procedure

1. Create the **/etc/dovecot/conf.d/10-replication.conf** file and perform the following steps in it:

a. Enable the **notify** and **replication** plug-ins:

```
mail_plugins = $mail_plugins notify replication
```

b. Add a **service replicator** section:

```
service replicator {
  process_min_avail = 1

  unix_listener replicator-doveadm {
    mode = 0600
    user = vmail
  }
}
```

With these settings, Dovecot starts at least one replicator process when the **dovecot** service starts. Additionally, this section defines the settings on the **replicator-doveadm** socket.

c. Add a **service aggregator** section to configure the **replication-notify-fifo** pipe and **replication-notify** socket:

```
service aggregator {
  fifo_listener replication-notify-fifo {
    user = vmail
  }
  unix_listener replication-notify {
    user = vmail
  }
}
```

d. Add a **service doveadm** section to define the port of the replication service:

```
service doveadm {
  inet_listener {
    port = 12345
  }
}
```

e. Set the password of the **doveadm** replication service:

```
doveadm_password = replication_password
```

The password must be the same on both servers.

f. Configure the replication partner:

```
plugin {
  mail_replica = tcp:server2.example.com:12345
}
```

g. Optional: Define the maximum number of parallel **dsync** processes:

```
replication_max_conns = 20
```

The default value of **replication_max_conns** is **10**.

2. Set secure permissions on the **/etc/dovecot/conf.d/10-replication.conf** file:

```
# chown root:root /etc/dovecot/conf.d/10-replication.conf
# chmod 600 /etc/dovecot/conf.d/10-replication.conf
```

3. Enable the **nis_enabled** SELinux Boolean to allow Dovecot to open the **doveadm** replication port:

```
setsebool -P nis_enabled on
```

4. Configure **firewalld** rules to allow only the replication partner to access the replication port, for example:

```
# firewall-cmd --permanent --zone=public --add-rich-rule="rule family="ipv4" source
address="192.0.2.1/32" port protocol="tcp" port="12345" accept"
# firewall-cmd --permanent --zone=public --add-rich-rule="rule family="ipv6" source
address="2001:db8:2::1/128" port protocol="tcp" port="12345" accept"
# firewall-cmd --reload
```

The subnet masks **/32** for the IPv4 and **/128** for the IPv6 address limit the access to the specified addresses.

5. Perform this procedure also on the other replication partner.

6. Reload Dovecot:

```
# systemctl reload dovecot
```

Verification

1. Perform an action in a mailbox on one server and then verify if Dovecot has replicated the change to the other server.

2. Display the replicator status:

```
# doveadm replicator status
Queued 'sync' requests       0
Queued 'high' requests       0
Queued 'low' requests        0
Queued 'failed' requests     0
Queued 'full resync' requests 30
Waiting 'failed' requests    0
Total number of known users  75
```

3. Display the replicator status of a specific user:

```
# doveadm replicator status example_user
username        priority  fast sync  full sync  success sync  failed
example_user    none      02:05:28   04:19:07   02:05:28      -
```

Additional resources

- **dsync(1)** man page
- **/usr/share/doc/dovecot/wiki/Replication.txt**

1.5. AUTOMATICALLY SUBSCRIBING USERS TO IMAP MAILBOXES

Typically, IMAP server administrators want Dovecot to automatically create certain mailboxes, such as **Sent** and **Trash**, and subscribe the users to them. You can set this in the configuration files.

Additionally, you can define *special-use mailboxes*. IMAP clients often support defining mailboxes for special purposes, such as for sent emails. To avoid that the user has to manually select and set the correct mailboxes, IMAP servers can send a **special-use** attribute in the IMAP **LIST** command. Clients can then use this attribute to identify and set, for example, the mailbox for sent emails.

Prerequisites

- Dovecot is configured.

Procedure

1. Update the **inbox** namespace section in the **/etc/dovecot/conf.d/15-mailboxes.conf** file:
   a. Add the **auto = subscribe** setting to each special-use mailbox that should be available to users, for example:

```
namespace inbox {
  ...
  mailbox Drafts {
    special_use = \Drafts
    auto = subscribe
  }

  mailbox Junk {
    special_use = \Junk
    auto = subscribe
  }

  mailbox Trash {
    special_use = \Trash
    auto = subscribe
  }

  mailbox Sent {
    special_use = \Sent
    auto = subscribe
  }
  ...
}
```

   If your mail clients support more special-use mailboxes, you can add similar entries. The **special_use** parameter defines the value that Dovecot sends in the **special-use** attribute to the clients.

   b. Optional: If you want to define other mailboxes that have no special purpose, add **mailbox** sections for them in the user's inbox, for example:

```
namespace inbox {
  ...
  mailbox "Important Emails" {
    auto = <value>
  }
  ...
}
```

You can set the **auto** parameter to one of the following values:

- **subscribe**: Automatically creates the mailbox and subscribes the user to it.
- **create**: Automatically creates the mailbox without subscribing the user to it.
- **no** (default): Dovecot neither creates the mailbox nor does it subscribe the user to it.

2. Reload Dovecot:

```
# systemctl reload dovecot
```

Verification

- Use an IMAP client and access your mailbox.
  Mailboxes with the setting **auto = subscribe** are automatically visible. If the client supports special-use mailboxes and the defined purposes, the client automatically uses them.

Additional resources

- RFC 6154: IMAP LIST Extension for Special-Use Mailboxes
- **/usr/share/doc/dovecot/wiki/MailboxSettings.txt**

1.6. CONFIGURING AN LMTP SOCKET AND LMTPS LISTENER

SMTP servers, such as Postfix, use the Local Mail Transfer Protocol (LMTP) to deliver emails to Dovecot. If the SMTP server runs:

- On the same host as Dovecot, use an LMTP socket
- On a different host, use an LMTP service
  By default, the LMTP protocol is not encrypted. However, if you configured TLS encryption, Dovecot uses the same settings automatically for the LMTP service. SMTP servers can then connect to it using the LMTPS protocol or the **STARTTLS** command over LMTP.

Prerequisites

- Dovecot is installed.
- If you want to configure an LMTP service, TLS encryption is configured in Dovecot.

Procedure

1. Verify that the LMTP protocol is enabled:

```
# doveconf -a | egrep "^protocols"
protocols = imap pop3 lmtp
```

The protocol is enabled, if the output contains **lmtp**.

2. If the **lmtp** protocol is disabled, edit the **/etc/dovecot/dovecot.conf** file, and append **lmtp** to the values in the **protocols** parameter:

```
protocols = ... lmtp
```

3. Depending on whether you need an LMTP socket or service, make the following changes in the **service lmtp** section in the **/etc/dovecot/conf.d/10-master.conf** file:

   - LMTP socket: By default, Dovecot automatically creates the **/var/run/dovecot/lmtp** socket. Optional: Customize the ownership and permissions:

```
service lmtp {
  ...
  unix_listener lmtp {
    mode = 0600
    user = postfix
    group = postfix
  }
  ...
}
```

   - LMTP service: Add a **inet_listener** sub-section:

```
service lmtp {
  ...
  inet_listener lmtp {
    port = 24
  }
  ...
}
```

4. Configure **firewalld** rules to allow only the SMTP server to access the LMTP port, for example:

```
# firewall-cmd --permanent --zone=public --add-rich-rule="rule family="ipv4" source
address="192.0.2.1/32" port protocol="tcp" port="24" accept"
# firewall-cmd --permanent --zone=public --add-rich-rule="rule family="ipv6" source
address="2001:db8:2::1/128" port protocol="tcp" port="24" accept"
# firewall-cmd --reload
```

The subnet masks **/32** for the IPv4 and **/128** for the IPv6 address limit the access to the specified addresses.

5. Reload Dovecot:

```
# systemctl reload dovecot
```

Verification

1. If you configured the LMTP socket, verify that Dovecot has created the socket and that the permissions are correct:

```
# ls -l /var/run/dovecot/lmtp
srw-------. 1 postfix postfix 0 Nov 22 17:17 /var/run/dovecot/lmtp
```

2. Configure the SMTP server to submit emails to Dovecot using the LMTP socket or service. When you use the LMTP service, ensure that the SMTP server uses the LMTPS protocol or sends the **STARTTLS** command to use an encrypted connection.

Additional resources

- **/usr/share/doc/dovecot/wiki/LMTP.txt**

1.7. DISABLING THE IMAP OR POP3 SERVICE IN DOVECOT

By default, Dovecot provides IMAP and POP3 services. If you require only one of them, you can disable the other to reduce the surface for attack.

Prerequisites

- Dovecot is installed.

Procedure

1. Uncomment the **protocols** parameter in the **/etc/dovecot/dovecot.conf** file, and set it to use the required protocols. For example, if you do not require POP3, set:

```
protocols = imap lmtp
```

   By default, the **imap**, **pop3**, and **lmtp** protocols are enabled.

2. Reload Dovecot:

```
# systemctl reload dovecot
```

3. Close the ports that are no longer required in the local firewall. For example, to close the ports for the POP3S and POP3 protocols, enter:

```
# firewall-cmd --remove-service=pop3s --remove-service=pop3
# firewall-cmd --reload
```

Verification

- Display all ports in **LISTEN** mode opened by the **dovecot** process:

```
# ss -tulp | grep dovecot
tcp  LISTEN 0  100  0.0.0.0:993  0.0.0.0:*  users:(("dovecot",pid=1405,fd=44))
tcp  LISTEN 0  100  0.0.0.0:143  0.0.0.0:*  users:(("dovecot",pid=1405,fd=42))
tcp  LISTEN 0  100     [::]:993     [::]:*  users:(("dovecot",pid=1405,fd=45))
tcp  LISTEN 0  100     [::]:143     [::]:*  users:(("dovecot",pid=1405,fd=43))
```

   In this example, Dovecot listens only on the TCP ports **993** (IMAPS) and **143** (IMAP).

Note that Dovecot only opens a port for the LMTP protocol if you configure the service to listen on a port instead of using a socket.

Additional resources

- **firewall-cmd(1)** man page

1.8. ENABLING SERVER-SIDE EMAIL FILTERING USING SIEVE ON A DOVECOT IMAP SERVER

You can upload Sieve scripts to a server using the ManageSieve protocol. Sieve scripts define rules and actions that a server should validate and perform on incoming emails. For example, users can use Sieve to forward emails from a specific sender, and administrators can create a global filter to move mails flagged by a spam filter into a separate IMAP folder.

The **ManageSieve** plugin adds support for Sieve scripts and the ManageSieve protocol to a Dovecot IMAP server.

WARNING

Use only clients that support using the ManageSieve protocol over TLS connections. Disabling TLS for this protocol causes clients to send credentials in plain text over the network.

Prerequisites

- Dovecot is configured and provides IMAP mailboxes.
- TLS encryption is configured in Dovecot.
- The mail clients support the ManageSieve protocol over TLS connections.

Procedure

1. Install the **dovecot-pigeonhole** package:

```
# dnf install dovecot-pigeonhole
```

2. Uncomment the following line in **/etc/dovecot/conf.d/20-managesieve.conf** to enable the **sieve** protocol:

```
protocols = $protocols sieve
```

   This setting activates Sieve in addition to the other protocols that are already enabled.

3. Open the ManageSieve port in **firewalld**:

```
# firewall-cmd --permanent --add-service=managesieve
# firewall-cmd --reload
```

4. Reload Dovecot:

```
# systemctl reload dovecot
```

Verification

1. Use a client and upload a Sieve script. Use the following connection settings:
   - Port: 4190
   - Connection security: SSL/TLS
   - Authentication method: PLAIN
2. Send an email to the user who has the Sieve script uploaded. If the email matches the rules in the script, verify that the server performs the defined actions.

Additional resources

- **/usr/share/doc/dovecot/wiki/Pigeonhole.Sieve.Plugins.IMAPSieve.txt**
- **/usr/share/doc/dovecot/wiki/Pigeonhole.Sieve.Troubleshooting.txt**
- **firewall-cmd(1)** man page

1.9. HOW DOVECOT PROCESSES CONFIGURATION FILES

The **dovecot** package provides the main configuration file **/etc/dovecot/dovecot.conf** and multiple configuration files in the **/etc/dovecot/conf.d/** directory. Dovecot combines the files to build the configuration when you start the service.

The main benefit of multiple config files is to group settings and increase readability. If you prefer a single configuration file, you can instead maintain all settings in **/etc/dovecot/dovecot.conf** and remove all **include** and **include_try** statements from that file.

Additional resources

- **/usr/share/doc/dovecot/wiki/ConfigFile.txt**
- **/usr/share/doc/dovecot/wiki/Variables.txt**

CHAPTER 2. DEPLOYING AND CONFIGURING A POSTFIX SMTP SERVER

As a system administrator, you can configure your email infrastructure by using a mail transport agent (MTA), such as Postfix, to transport email messages between hosts using the SMTP protocol. Postfix is a server-side application for routing and delivering mail. You can use Postfix to set up a local mail server, create a null-client mail relay, use a Postfix server as a destination for multiple domains, or choose an LDAP directory instead of files for lookups.

The key features of Postfix:

- Security features to protect against common email related threats
- Customization options, including support for virtual domains and aliases

2.1. OVERVIEW OF THE MAIN POSTFIX CONFIGURATION FILES

The **postfix** package provides multiple configuration files in the **/etc/postfix/** directory.

To configure your email infrastructure, use the following configuration files:

- **main.cf** – contains the global configuration of Postfix.
- **master.cf** – specifies Postfix interaction with various processes to accomplish mail delivery.
- **access** – specifies access rules, for example hosts that are allowed to connect to Postfix.
- **transport** – maps email addresses to relay hosts.
- **aliases** – contains a configurable list required by the mail protocol that describes user ID aliases. Note that you can find this file in the **/etc/** directory.

2.2. INSTALLING AND CONFIGURING A POSTFIX SMTP SERVER

You can configure your Postfix SMTP server to receive, store, and deliver email messages. If the mail server package is not selected during the system installation, Postfix will not be available by default. Perform the following steps to install Postfix:

Prerequisites

- You have the root access.
- Register your system.
- Disable and remove Sendmail:

```
# dnf remove sendmail
```

Procedure

1. Install Postfix:

```
# dnf install postfix
```

2. To configure Postfix, edit the **/etc/postfix/main.cf** file and make the following changes:

   a. By default, Postfix receives emails only on the **loopback** interface. To configure Postfix to listen on specific interfaces, update the **inet_interfaces** parameter to the IP addresses of these interfaces:

   ```
   inet_interfaces = 127.0.0.1/32, [::1]/128, 192.0.2.1, [2001:db8:1::1]
   ```

   To configure Postfix to listen on all interfaces, set:

   ```
   inet_interfaces = all
   ```

   b. If you want that Postfix uses a different hostname than the fully-qualified domain name (FQDN) that is returned by the **gethostname()** function, add the **myhostname** parameter:

   ```
   myhostname = <smtp.example.com>
   ```

   For example, Postfix adds this hostname to header of emails it processes.

   c. If the domain name differs from the one in the **myhostname** parameter, add the **mydomain** parameter:

   ```
   mydomain = <example.com>
   ```

   d. Add the **myorigin** parameter and set it to the value of **mydomain**:

   ```
   myorigin = $mydomain
   ```

   With this setting, Postfix uses the domain name as origin for locally posted mails instead of the hostname.

   e. Add the **mynetworks** parameter, and define the IP ranges of trusted networks that are allowed to send mails:

   ```
   mynetworks = 127.0.0.1/32, [::1]/128, 192.0.2.1/24, [2001:db8:1::1]/64
   ```

   If clients from not trustworthy networks, such as the Internet, should be able to send mails through this server, you must configure relay restrictions in a later step.

3. Verify if the Postfix configuration in the **main.cf** file is correct:

   ```
   $ postfix check
   ```

4. Enable the **postfix** service to start at boot and start it:

   ```
   # systemctl enable --now postfix
   ```

5. Allow the smtp traffic through firewall and reload the firewall rules:

   ```
   # firewall-cmd --permanent --add-service smtp

   # firewall-cmd --reload
   ```

Verification

1. Verify that the **postfix** service is running:

```
# systemctl status postfix
```

   - Optional: Restart the **postfix** service, if the output is stopped, waiting, or the service is not running:

```
# systemctl restart postfix
```

   - Optional: Reload the **postfix** service after changing any options in the configuration files in the **/etc/postfix/** directory to apply those changes:

```
# systemctl reload postfix
```

2. Verify the email communication between local users on your system:

```
# echo "This is a test message" | mail -s <SUBJECT> <user@mydomain.com>
```

3. Verify that your mail server is not an open relay by sending an email to an email address outside of your domain from a client (**server1**) to your mail server (**server2**):

   a. Edit the **/etc/postfix/main.cf** file for **server1** as follows:

```
relayhost = <ip_address_of_server2>
```

   b. Edit the **/etc/postfix/main.cf** file for **server2** as follows:

```
mynetworks = <ip_address_of_server2>
```

   c. On **server1**, send the following mail:

```
# echo "This is an open relay test message" | mail -s <SUBJECT>
<user@example.com>
```

   d. Check the **/var/log/maillog** file:

```
554 Relay access denied - the server is not going to relay.
250 OK or similar - the server is going to relay.
```

Troubleshooting

- In case of errors, check the **/var/log/maillog** file.

Additional resources

- The **/etc/postfix/main.cf** configuration file
- The **/usr/share/doc/postfix/README_FILES** directory
- Using and configuring firewalld

2.3. CUSTOMIZING TLS SETTINGS OF A POSTFIX SERVER

To make your email traffic encrypted and therefore more secure, you can configure Postfix to use a certificate from a trusted certificate authority (CA) instead of the self-signed certificate and customize the Transport Layer Security (TLS) security settings. In RHEL 9, the TLS encryption protocol is enabled in the Postfix server by default. The basic Postfix TLS configuration contains self-signed certificates for inbound SMTP and the opportunistic TLS for outbound SMTP.

Prerequisites

- You have the root access.
- You have the **postfix** package installed on your server.
- You have a certificate signed by a trusted certificate authority (CA) and a private key.
- You have copied the following files to the Postfix server:
  - The server certificate: **/etc/pki/tls/certs/postfix.pem**
  - The private key: **/etc/pki/tls/private/postfix.key**
- If the server runs RHEL 9.2 or later and the FIPS mode is enabled, clients must either support the Extended Master Secret (EMS) extension or use TLS 1.3. TLS 1.2 connections without EMS fail. For more information, see the TLS extension "Extended Master Secret" enforced Knowledgebase article.

Procedure

1. Set the path to the certificate and private key files on the server where Postfix is running by adding the following lines to the **/etc/postfix/main.cf** file:

```
smtpd_tls_cert_file = /etc/pki/tls/certs/postfix.pem
smtpd_tls_key_file = /etc/pki/tls/private/postfix.key
```

2. Restrict the incoming SMTP connections to authenticated users only by editing the **/etc/postfix/main.cf** file:

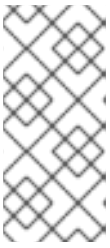
```
smtpd_tls_auth_only = yes
```

3. Reload the **postfix** service to apply the changes:

```
# systemctl reload postfix
```

Verification

- Configure your client to use TLS encryption and send an email.



NOTE

To get additional information about Postfix client TLS activity, increase the log level from **0** to **1** by changing the following line in the **/etc/postfix/main.cf**:

```
smtp_tls_loglevel = 1
```

2.4. CONFIGURING POSTFIX TO FORWARD ALL EMAILS TO A MAIL RELAY

If you want to forward all email to a mail relay, you can configure Postfix server as a null client. In this configuration Postfix only forwards mail to a different mail server and is not capable of receiving mail.

Prerequisites

- You have the root access.
- You have the **postfix** package installed on your server.
- You have the IP address or hostname of the relay host to which you want to forward emails.

Procedure

1. To prevent Postfix from accepting any local email delivery and making it a null client, edit the **/etc/postfix/main.cf** file and make the following changes:
   a. Configure Postfix to forward all email by setting the **mydestination** parameter equal to an empty value:

      ```
      mydestination =
      ```

      In this configuration the Postfix server is not a destination for any email and acts as a null client.

   b. Specify the mail relay server that receives the email from your null client:

      ```
      relayhost = <[ip_address_or_hostname]>
      ```

      The relay host is responsible for the mail delivery. Enclose ***<ip_address_or_hostname>*** in square brackets.

   c. Configure the Postfix mail server to listen only on the loopback interface for emails to deliver:

      ```
      inet_interfaces = loopback-only
      ```

   d. If you want Postfix to rewrite the sender domain of all outgoing emails to the company domain of your relay mail server, set:

      ```
      myorigin = <relay.example.com>
      ```

   e. To disable the local mail delivery, add the following directive at the end of the configuration file:

      ```
      local_transport = error: local delivery disabled
      ```

   f. Add the **mynetworks** parameter so that Postfix forwards email from the local system originating from the 127.0.0.0/8 IPv4 network and the [::1]/128 IPv6 network to the mail relay server:

      ```
      mynetworks = 127.0.0.0/8, [::1]/128
      ```

-

2. Verify if the Postfix configuration in the **main.cf** file is correct:

```
$ postfix check
```

3. Restart the **postfix** service to apply the changes:

```
# systemctl restart postfix
```

Verification

- Verify that the email communication is forwarded to the mail relay:

```
# echo "This is a test message" | mail -s <SUBJECT> <user@example.com>
```

Troubleshooting

- In case of errors, check the /**var**/**log**/**maillog** file.

Additional resources

- The /**etc**/**postfix**/**main.cf** configuration file

2.5. CONFIGURING POSTFIX AS A DESTINATION FOR MULTIPLE DOMAINS

You can configure Postfix as a mail server that can receive emails for multiple domains. In this configuration, Postfix acts as the final destination for emails sent to addresses within the specified domains. You can configure the following:

- Set up multiple email addresses that point to the same email destination
- Route incoming email for multiple domains to the same Postfix server

Prerequisites

- You have the root access.
- You have configured a Postfix server.

Procedure

1. In the /**etc**/**postfix**/**virtual** virtual alias file, specify the email addresses for each domain. Add each email address on a new line:

```
<info@example.com>  <user22@example.net>
<sales@example.com>  <user11@example.org>
```

   In this example, Postfix redirects all emails sent to info@example.com to user22@example.net and email sent to sales@example.com to user11@example.org.

2. Create a hash file for the virtual alias map:

```
# postmap /etc/postfix/virtual
```

This command creates the **/etc/postfix/virtual.db** file. Note that you must always re-run this command after you update the **/etc/postfix/virtual** file.

3. In the Postfix **/etc/postfix/main.cf** configuration file, add the **virtual_alias_maps** parameter and point it to the hash file:

```
virtual_alias_maps = hash:/etc/postfix/virtual
```

4. Reload the **postfix** service to apply the changes:

```
# systemctl reload postfix
```

Verification

- Test the configuration by sending an email to one of the virtual email addresses.

Troubleshooting

- In case of errors, check the **/var/log/maillog** file.

2.6. USING AN LDAP DIRECTORY AS A LOOKUP TABLE

If you use a Lightweight Directory Access Protocol (LDAP) server to store accounts, domains or aliases, you can configure Postfix to use the LDAP server as a lookup table. Using LDAP instead of files for lookups enables you to have a central database.

Prerequisites

- You have the root access.
- You have the **postfix** package installed on your server.
- You have an LDAP server with the required schema and user credentials.
- You have the **postfix-ldap** plugin installed on the server running Postfix.

Procedure

1. Configure the LDAP lookup parameters by creating a **/etc/postfix/ldap-aliases.cf** file with the following content:

   a. Specify the hostname of the LDAP server:

   ```
   server_host = <ldap.example.com>
   ```

   b. Specify the base domain name for the LDAP search:

   ```
   search_base = dc=<example>,dc=<com>
   ```

   c. Optional: Customize the LDAP search filter and attributes based on your requirements. The filter for searching the directory defaults to **query_filter = mailacceptinggeneralid=%s**.

2. Enable the LDAP source as a lookup table in the **/etc/postfix/main.cf** configuration file by adding the following content:

```
virtual_alias_maps = ldap:/etc/postfix/ldap-aliases.cf
```

3. Verify the LDAP configuration by running the **postmap** command, which checks for any syntax errors or connectivity issues:

```
# postmap -q @<example.com> ldap:/etc/postfix/ldap-aliases.cf
```

4. Reload the **postfix** service to apply the changes:

```
# systemctl reload postfix
```

Verification

- Send a test email to verify that the LDAP lookup works correctly. Check the mail logs in **/var/log/maillog** for any errors.

Additional resources

- **/usr/share/doc/postfix/README_FILES/LDAP_README** file
- **/usr/share/doc/postfix/README_FILES/DATABASE_README** file

2.7. CONFIGURING POSTFIX AS AN OUTGOING MAIL SERVER TO RELAY FOR AUTHENTICATED USERS

You can configure Postfix to relay mail for authenticated users. In this scenario, you allow users to authenticate themselves and use their email address to send mail through your SMTP server by configuring Postfix as an outgoing mail server with SMTP authentication, TLS encryption, and sender address restrictions.

Prerequisites

- You have the root access.
- You have configured a Postfix server.

Procedure

1. To configure Postfix as an outgoing mail server, edit the **/etc/postfix/main.cf** file and add the following:

   a. Enable SMTP authentication:

```
smtpd_sasl_auth_enable = yes
broken_sasl_auth_clients = yes
```

   b. Disable access without TLS:

```
smtpd_tls_auth_only = yes
```

c. Allow mail relaying only for authenticated users:

```
smtpd_relay_restrictions = permit_mynetworks permit_sasl_authenticated
defer_unauth_destination
```

d. Optional: Restrict users to use their own email address only as a sender:

```
smtpd_sender_restrictions = reject_sender_login_mismatch
```

2. Reload the **postfix** service to apply the changes:

```
# systemctl reload postfix
```

Verification

- Authenticate in your SMTP client that supports TLS and SASL. Send an test email to verify that the SMTP authentication works correctly.

2.8. DELIVERING EMAIL FROM POSTFIX TO DOVECOT RUNNING ON THE SAME HOST

You can configure Postfix to deliver incoming mail to Dovecot on the same host using LMTP over a UNIX socket. This socket enables direct communication between Postfix and Dovecot on the local machine.

Prerequisites

- You have the root access.
- You have configured a Postfix server.
- You have configured a Dovecot server, see Configuring and maintaining a Dovecot IMAP and POP3 server.
- You have configured the LMTP socket on your Dovecot server, see Configuring an LMTP socket and LMTPS listener.

Procedure

1. Configure Postfix to use the LMTP protocol and the UNIX domain socket for delivering mail to Dovecot in the **/etc/postfix/main.cf** file:
   - If you want to use virtual mailboxes, add the following content:

```
virtual_transport = lmtp:unix:/var/run/dovecot/lmtp
```

   - If you want to use non-virtual mailboxes, add the following content:

```
mailbox_transport = lmtp:unix:/var/run/dovecot/lmtp
```

2. Reload **postfix** to apply the changes:

```
# systemctl reload postfix
```

Verification

- Send an test email to verify that the LMTP socket works correctly. Check the mail logs in **/var/log/maillog** for any errors.

2.9. DELIVERING EMAIL FROM POSTFIX TO DOVECOT RUNNING ON A DIFFERENT HOST

You can establish a secure connection between Postfix mail server and the Dovecot delivery agent over the network. To do so, configure the LMTP service to use network socket for delivering mail between mail servers. By default, the LMTP protocol is not encrypted. However, if you configured TLS encryption, Dovecot uses the same settings automatically for the LMTP service. SMTP servers can then connect to it using the **STARTTLS** command over LMTP.

Prerequisites

- You have the root access.
- You have configured a Postfix server.
- You have configured a Dovecot server, see Configuring and maintaining a Dovecot IMAP and POP3 server.
- You have configured the LMTP service on your Dovecot server, see Configuring an LMTP socket and LMTPS listener.

Procedure

1. Configure Postfix to use the LMTP protocol and the INET domain socket for delivering mail to Dovecot in the **/etc/postfix/main.cf** file by adding the following content:

```
mailbox_transport = lmtp:inet:<dovecot_host>:<port>
```

   Replace ***<dovecot_host>*** with the IP address or hostname of the Dovecot server and **<port>** with the port number of the LMTP service.

2. Reload the **postfix** service to apply the changes:

```
# systemctl reload postfix
```

Verification

- Send an test email to an address hosted by the remote Dovecot server and check the Dovecot logs to ensure that the mail was successfully delivered.

2.10. SECURING THE POSTFIX SERVICE

Postfix is a mail transfer agent (MTA) that uses the Simple Mail Transfer Protocol (SMTP) to deliver electronic messages between other MTAs and to email clients or delivery agents. Although MTAs can encrypt traffic between one another, they might not do so by default. You can also mitigate risks to various attacks by changing setting to more secure values.

2.10.1. Reducing Postfix network-related security risks

To reduce the risk of attackers invading your system through the network, perform as many of the following tasks as possible.

- Do not share the **/var/spool/postfix/** mail spool directory on a Network File System (NFS) shared volume. NFSv2 and NFSv3 do not maintain control over user and group IDs. Therefore, if two or more users have the same UID, they can receive and read each other's mail, which is a security risk.

NOTE

This rule does not apply to NFSv4 using Kerberos, because the **SECRPC_GSS** kernel module does not use UID-based authentication. However, to reduce the security risks, you should not put the mail spool directory on NFS shared volumes.

- To reduce the probability of Postfix server exploits, mail users must access the Postfix server using an email program. Do not allow shell accounts on the mail server, and set all user shells in the **/etc/passwd** file to **/sbin/nologin** (with the possible exception of the **root** user).

- To protect Postfix from a network attack, it is set up to only listen to the local loopback address by default. You can verify this by viewing the **inet_interfaces = localhost** line in the **/etc/postfix/main.cf** file. This ensures that Postfix only accepts mail messages (such as **cron** job reports) from the local system and not from the network. This is the default setting and protects Postfix from a network attack. To remove the localhost restriction and allow Postfix to listen on all interfaces, set the **inet_interfaces** parameter to **all** in **/etc/postfix/main.cf**.

2.10.2. Postfix configuration options for limiting DoS attacks

An attacker can flood the server with traffic, or send information that triggers a crash, causing a denial of service (DoS) attack. You can configure your system to reduce the risk of such attacks by setting limits in the **/etc/postfix/main.cf** file. You can change the value of the existing directives or you can add new directives with custom values in the *<directive> = <value>* format.

Use the following list of directives for limiting a DoS attack:

smtpd_client_connection_rate_limit

This directive limits the maximum number of connection attempts any client can make to this service per time unit. The default value is **0**, which means a client can make as many connections per time unit as Postfix can accept. By default, the directive excludes clients in trusted networks.

anvil_rate_time_unit

This directive is a time unit to calculate the rate limit. The default value is **60** seconds.

smtpd_client_event_limit_exceptions

This directive excludes clients from the connection and rate limit commands. By default, the directive excludes clients in trusted networks.

smtpd_client_message_rate_limit

This directive defines the maximum number of message deliveries from client to request per time unit (regardless of whether or not Postfix actually accepts those messages).

default_process_limit

This directive defines the default maximum number of Postfix child processes that provide a given service. You can ignore this rule for specific services in the **master.cf** file. By default, the value is **100**.

queue_minfree

This directive defines the minimum amount of free space required to receive mail in the queue file system. The directive is currently used by the Postfix SMTP server to decide if it accepts any mail at all. By default, the Postfix SMTP server rejects **MAIL FROM** commands when the amount of free space is less than 1.5 times the **message_size_limit**. To specify a higher minimum free space limit, specify a **queue_minfree** value that is at least 1.5 times the **message_size_limit**. By default, the **queue_minfree** value is **0**.

header_size_limit

This directive defines the maximum amount of memory in bytes for storing a message header. If a header is large, it discards the excess header. By default, the value is **102400** bytes.

message_size_limit

This directive defines the maximum size of a message including the envelope information in bytes. By default, the value is **10240000** bytes.

2.10.3. Configuring Postfix to use SASL

Postfix supports Simple Authentication and Security Layer (SASL) based SMTP Authentication (AUTH). SMTP AUTH is an extension of the Simple Mail Transfer Protocol. Currently, the Postfix SMTP server supports the SASL implementations in the following ways:

Dovecot SASL

The Postfix SMTP server can communicate with the Dovecot SASL implementation using either a UNIX-domain socket or a TCP socket. Use this method if Postfix and Dovecot applications are running on separate machines.

Cyrus SASL

When enabled, SMTP clients must authenticate with the SMTP server using an authentication method supported and accepted by both the server and the client.

Prerequisites

- The **dovecot** package is installed on the system

Procedure

1. Set up Dovecot:

   a. Include the following lines in the **/etc/dovecot/conf.d/10-master.conf** file:

   ```
   service auth {
     unix_listener /var/spool/postfix/private/auth {
       mode = 0660
       user = postfix
       group = postfix
     }
   }
   ```

   The previous example uses UNIX-domain sockets for communication between Postfix and Dovecot. The example also assumes default Postfix SMTP server settings, which include the mail queue located in the **/var/spool/postfix/** directory, and the application running under the **postfix** user and group.

   b. Optional: Set up Dovecot to listen for Postfix authentication requests through TCP:

```
service auth {
  inet_listener {
      port = port-number
  }
}
```

c. Specify the method that the email client uses to authenticate with Dovecot by editing the **auth_mechanisms** parameter in **/etc/dovecot/conf.d/10-auth.conf** file:

```
auth_mechanisms = plain login
```

The **auth_mechanisms** parameter supports different plaintext and non-plaintext authentication methods.

2. Set up Postfix by modifying the **/etc/postfix/main.cf** file:

a. Enable SMTP Authentication on the Postfix SMTP server:

```
smtpd_sasl_auth_enable = yes
```

b. Enable the use of Dovecot SASL implementation for SMTP Authentication:

```
smtpd_sasl_type = dovecot
```

c. Provide the authentication path relative to the Postfix queue directory. Note that the use of a relative path ensures that the configuration works regardless of whether the Postfix server runs in **chroot** or not:

```
smtpd_sasl_path = private/auth
```

This step uses UNIX-domain sockets for communication between Postfix and Dovecot.

To configure Postfix to look for Dovecot on a different machine in case you use TCP sockets for communication, use configuration values similar to the following:

```
smtpd_sasl_path = inet: ip-address : port-number
```

In the previous example, replace the *ip-address* with the IP address of the Dovecot machine and *port-number* with the port number specified in Dovecot's **/etc/dovecot/conf.d/10-master.conf** file.

d. Specify SASL mechanisms that the Postfix SMTP server makes available to clients. Note that you can specify different mechanisms for encrypted and unencrypted sessions.

```
smtpd_sasl_security_options = noanonymous, noplaintext
smtpd_sasl_tls_security_options = noanonymous
```

The previous directives specify that during unencrypted sessions, no anonymous authentication is allowed and no mechanisms that transmit unencrypted user names or passwords are allowed. For encrypted sessions that use TLS, only non-anonymous authentication mechanisms are allowed.

Additional resources

- Postfix SMTP server policy - SASL mechanism properties
- Postfix and Dovecot SASL
- Configuring SASL authentication in the Postfix SMTP server