NixOS Mailserver

NixOS Mailserver Contributors

Jun 19, 2026

CONTENTS


SNM
Simple Nixos Mailserver

CHAPTER

ONE

SETUP GUIDE

Mail servers can be a tricky thing to set up. This guide is supposed to run you through the most important steps to achieve a 10/10 score on https://mail-tester.com.

1.1 Requirements

To set up a self-hosted mail server, you need the following:

- Small (e.g. 1C/2G) server running NixOS
- Stable IPv4 and - strongly recommended - IPv6 addresses
  - Ability to configure a Reverse DNS (PTR record) for your IP addresses
  - Access to SMTP traffic on port 25/tcp - some hosters make you ask for this
- A registered domain name with DNS record management access

Once these requirements are in place, you can begin setting up your selfhosted mailserver.

Note: Below we'll assume that your server got assigned the public IP addresses `192.0.2.1` (IPv4) and `2001:db8::1` (IPv6) and that you control the `example.com` domain.

1.2 Configure forward DNS records

Here we set up `mail.example.com` as the forward hostname for your mail server to point to the IP addresses allocated to the server. This allows reaching the server under this name and to reference it later in MX records for mail delivery.

Now edit the `example.com` zone and create the following DNS records:

Name	TTL	Type	Value
mail.example.com.	3600	A	192.0.2.1
mail.example.com.	3600	AAAA	2001:db8::1

Note: If your server does not have an IPv6 address, you must skip the `AAAA` record.

1.3 Verify DNS record propagation

Before we continue with the next step, we require that the forward DNS record has propagated. For that it's best to check an authoritative nameserver for `example.com` so that we don't look at cached DNS records.

```
# Find the authoritative nameservers for example.com
$ nix-shell -p dig --command "dig NS example.com +short"
ns1.example.org.
ns2.example.org.

# Query the A record from an authoritative nameserver
$ nix-shell -p dig --command "dig @ns1.example.org A mail.example.com +short"
192.0.2.1

# Query the AAAA record from an authoritative nameserver
$ nix-shell -p dig --command "dig @ns1.example.org AAAA mail.example.com +short"
2001:db8::1
```

DNS propagation usually takes a few minutes, so you might need to retry these queries. Once the IP addresses appear you can continue with the next step.

1.4 Setup the server

The following configuration describes a fairly complete mail server, capable of sending and receiving mail for statically configured accounts. It includes encrypted SMTP and IMAP services for secure delivery and retrieval, and relies on ACME HTTP-01 to automatically obtain and maintain a TLS certificate.

While more options are available, the configuration below covers the most common settings to get your mail server up and running.

```
{
  config,
  ...
}:
{
  imports = [
    (builtins.fetchTarball {
      # This is a quick and dirty way to import a NixOS mailserver release. What
      # you should do long-term is use a proper dependency pinning tool like npins
      # or flakes.

      # URL to the tarball for the release matching your NixOS release
      url = "https://gitlab.com/simple-nixos-mailserver/nixos-mailserver/-/archive/nixos-
↪26.05/nixos-mailserver-nixos-26.05.tar.gz";

      # Hash of the unpacked tarball, run the following command to retrieve it
      # release="nixos-26.05" nix-prefetch-url "https://gitlab.com/simple-nixos-
↪mailserver/nixos-mailserver/-/archive/${release}/nixos-mailserver-${release}.tar.gz" --
↪unpack
      sha256 = "0000000000000000000000000000000000000000000000000000";
    })
  ];
```

(continues on next page)

(continued from previous page)

```
  # https://letsencrypt.org/repository/#let-s-encrypt-subscriber-agreement
  security.acme.acceptTerms = true;

  # Allow incoming HTTP connections
  networking.firewall.allowedTCPPorts = [ 80 ];

  # Enable ACME HTTP-01 challenge with nginx
  services.nginx = {
    enable = true;
    virtualHosts.${config.mailserver.fqdn}.enableACME = true;
  };

  mailserver = {
    enable = true;
    stateVersion = 5;
    fqdn = "mail.example.com";
    domains = [ "example.com" ];

    # Reference the existing ACME configuration created by nginx
    x509.useACMEHost = config.mailserver.fqdn;

    # A list of all login accounts. To create the password hashes, use
    # nix-shell -p mkpasswd --run 'mkpasswd -s'
    accounts = {
      "user1@example.com" = {
        # Reads the password hash from a file on the server
        hashedPasswordFile = "/a/file/containing/a/hashed/password";

        # Additional addresses delivered to this mailbox
        aliases = [ "postmaster@example.com" ];
      };
      "user2@example.com" = {
        # Provides the password hash inline
        hashedPassword = "$y$j9T$JqqefR6flaaJBRjD4KVZc1$QM6h4Spr5.yn/FuIT.
↪ydTV22daEbiVd8ZprV/POtPgB";
      };
    };
  };
}
```

After a `nixos-rebuild switch` your server should be running all the necessary mail services.

1.5 Configure DNS records

1.5.1 Reverse DNS

Earlier, we configured forward DNS from your hostname to your IP address. Now we will configure reverse DNS so that your IP address points back to your hostname.

If your forward and reverse DNS do not match, many mail servers will reject or flag your emails as spam, severely impairing delivery.

Your server provider should allow you to configure reverse DNS (PTR record) records for the IP addresses you control, typically through their control panel or account management interface:

- Configure `192.0.2.1` to point to `mail.example.com.`
- Configure `2001:db8::1` to point to `mail.example.com.`, if you have IPv6 addressing

Alternatively, if you interact with a proper DNS setup, the actual DNS records look like this:

Name	TTL	Type	Value
1.2.0.192.in-addr.arpa.	86400	PTR	mail.example.com.
1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.8.b.d.0.1.0.0.2.ip6.arpa.	86400	PTR	mail.example.com.

Note: Reverse DNS uses reverse notation for naming its records:

- IPv4 reverses the order of the octets and appends `in-addr.arpa.`, so `192.0.2.1` becomes `1.2.0.192.in-addr.arpa.`
- IPv6 fully expands the address and reverses each hex digit before concatenating it with dots and appending `ip6.arpa.`

```
nix-shell -p haskellPackages.ip6addr --command "ip6addr --ptr 2001:db8::1"
1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.8.b.d.0.1.0.0.2.IP6.ARPA.
```

Warning: We don't recommend setting up a mail server if you are unable to configure reverse DNS on your public IP addresses because mails would inevitable be marked as spam. Note that many residential ISP providers don't allow you to set a reverse DNS entry and prohibit sending mail through policy blocklists like Spamhaus PBL.

DNS propagation often isn't instant, so verify before continuing:

```
$ nix-shell -p dig --command "dig -x 192.0.2.1 +short"
mail.example.com.

$ nix-shell -p dig --command "dig -x 2001:db8::1 +short"
mail.example.com.
```

1.5.2 MX record

The MX record instructs other mailservers where to deliver mail for a domain name.

Create the MX record for `example.com` to point to the hostname of the server.

Name	TTL	Priority	Type	Value
example.com.	3600	MX	10	mail.example.com.

The priority field determines the order when multiple servers are configured. It is not important in this scenario but setting a value is mandatory and 10 leaves some wiggle room below and above, should you ever need that.

```
$ nix-shell -p dig --command "dig @ns1.example.org MX example.com +short"
10 mail.example.com.
```

1.5.3 SPF record

With SPF we can specify which mail servers are authorized to send mail on behalf of a domain name.

The SPF record is TXT record and we can tie it in with the MX record we created in the previous step. Finishing with `-all` indicates that without any match the mail should be rejected.

Name	TTL	Type	Value
example.com.	86400	TXT	v=spf1 mx -all

```
$ nix-shell -p dig --command "dig TXT example.com +short"
v=spf1 mx -all
```

1.5.4 DKIM record

On system activation a DKIM keypair for `example.com` was generated. The mail server uses this key to sign outgoing emails, allowing receiving servers to verify the authenticity of the sender domain and ensuring that the signed parts of the message have not been tampered with.

Now, check `/var/dkim/example.com.mail.txt`, which contains the proposed DNS record for the `mail` DKIM selector.

```
mail._domainkey IN TXT ( "v=DKIM1; k=rsa; "
 "p=MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAv7hSess/UgEjaaq/NDn5KtW2iZzYljhf45DH3tN/"
↪kqcJ04JJk/Z1rS7CMJQ/pYZSSnQOju0H25uOtODvhqXPDxDdtCyDSrx54z/38lGNtA76/iWy/
↪ikjb9hEkb2k3HuKex3P4KhhOC1pytDEFnh/T2aBxPNOigc/
↪cpqm1U9RbnAwvArtx9dgOAgiV8rOIgPgyrPw1B3cJG3hgFYU2"
 "GwXMoiFQPgwm7bkjelmThqXozA7jFJfnYt49jjrIYCv8X/nQx9cNpVAv2852mhU/
↪3uuy6sa4MPjT6RiK9BJCMyDnqSpTPCjIubL4VhGCuzp7RPBkayWnlaH0X8PWGq6BQ0eBwIDAQAB"
) ;
```

Based on the content of this file, we can create the DKIM TXT record for the `mail` selector in the `example.com` zone. For the `p=` value, glue the two long strings back together without any quotes and spaces and put them into your record below.

Name	TTL	Type	Value
mail._domainkey.example.com.	86400	TXT	v=DKIM1; k=rsa; p=MIIBIjANBgk...Q0eBwIDAQAB

```
$ nix-shell -p dig --command "dig @ns1.example.org TXT mail._domainkey.example.com +short
↪"
"v=DKIM1; k=rsa; p=MIIBIjANBgk...Q0eBwIDAQAB"
```

1.5.5 DMARC record

Finally, DMARC lets you define a policy for how strictly SPF and DKIM should be checked and how to handle validation failures. For a new server, it's important to have a DMARC record in place, even if it doesn't enforce any actions yet, because it improves deliverability by showing receiving servers that your domain is properly managed and reducing the risk of email spoofing.

Name	TTL	Type	Value
_dmarc.example.com.	86400	TXT	v=DMARC1; p=none;

Verify propagation one final time.

```
$ nix-shell -p dig --command "dig @ns1.example.org TXT _dmarc.example.com +short"
"v=DMARC1; p=none"
```

1.6 Test your Setup

Write an email to your aunt — she's been waiting far too long for your reply, and this is your chance to finally make her day. Or, if you prefer a less emotional test, send a message to mail-tester.com to see how your outgoing mail scores.

You can also let MXToolbox take a peek at your setup. If you followed the steps carefully, everything should be working perfectly!

1.7 Join the community

The community has a lively chat room on Matrix at #nixos-mailserver:nixos.org where you can ask questions, get help, share ideas, or discuss contributions.

1.8 Next steps

Your server scored perfect results already, so these steps are entirely optional.

Are you feeling adventurous? Dive into our advanced configurations to explore additional features and capabilities that let you fine-tune and extend your mail setup.

If you want to take things even further, more elaborate testing services can give you a clearer picture of your mail service and suggest ways to improve it.

- internet.nl (supported by the Dutch Government)
- MECSA (supported by the European Commission)

Finally, you can also browse the full list of options provided by NixOS mailserver.

CHAPTER

TWO

ADVANCED CONFIGURATIONS

Congratulations on completing the Setup Guide!

If you're an experienced mailserver admin, then you probably know what you want to do next. Our How-to guides (accessible in the navigation sidebar) might help you accomplish your goals. If not, consider contributing a guide!

If this is your first mailserver, consider the following:

- Configure regular, automatic backups.
- Enable fulltext search to let clients that don't sync all mail efficiently find messages, by performing searches directly on the server.
- Set up the Sender Rewriting Scheme if you rely on server-side mail forwarding to external mail servers using mail forwards or Sieve rules.
- Contribute DMARC reports and SMTP TLS reports to help improve email security across the internet by sending feedback on authentication failures, spoofing attempts, and TLS encryption issues.

CHAPTER

THREE

CONTRIBUTE OR TROUBLESHOOT

To report an issue, please go to https://gitlab.com/simple-nixos-mailserver/nixos-mailserver/-/issues.

If you have questions, feel free to reach out:

- Matrix: #nixos-mailserver:nixos.org
- IRC: #nixos-mailserver on Libera Chat

All our workflows rely on Nix being configured with Flakes.

3.1 Licensing and copyright assignment

This project is licensed under GPL-3.0-or-later and uses SPDX headers to record licensing and copyright information.

We do not attempt to track copyright ownership based on individual commits or lines of code. Detailed authorship information remains available through the Git history.

Most files are covered by the default annotations in `REUSE.toml`, which assign the collective attribution "The NixOS Mailserver contributors" under `GPL-3.0-or-later`.

Files that are likely to be copied or reused independently, such as NixOS modules, tests, migration scripts and code, should additionally contain explicit SPDX headers:

```
SPDX-License-Identifier: GPL-3.0-or-later
SPDX-FileCopyrightText: The NixOS Mailserver contributors
```

For new files containing substantial original work, consider adding yourself as a copyright holder:

```
SPDX-License-Identifier: GPL-3.0-or-later
SPDX-FileCopyrightText: Your Name <you@example.com>
```

3.2 Development Shell

We provide a *flake.nix* devshell that automatically sets up pre-commit hooks, which allows for fast feedback cycles when making changes to the repository.

```
$ nix develop
```

We recommend setting up direnv to automatically attach to the development environment when entering the project directories.

3.3 Run NixOS tests

To run the test suite, you need to enable Nix Flakes.

You can then run the testsuite via

```
$ nix flake check -L
```

Since Nix doesn't guarantee your machine have enough resources to run all test VMs in parallel, some tests can fail. You would then have to run tests manually. For instance:

```
$ nix build .#checks.x86_64-linux.external-unstable -L
```

3.4 Contributing to the documentation

The documentation is written in RST (except option documentation which is in CommonMark), built with Sphinx and published by Read the Docs.

For the syntax, see the RST/Sphinx primer.

To build the documentation, you need to enable Nix Flakes.

```
$ nix build .#documentation
$ xdg-open result/index.html
```

3.5 Manual migrations

We need to take great care around providing a migration story around breaking changes. If manual intervention becomes necessary we provide the *stateVersion* option to notify the user that they need to complete a migration before they can deploy an update.

If that is the case for your change, find the highest *stateVersion* that is being asserted on in *mail-server/assertions.nix*. Then pick the next number and add a new assertion, write a good summary describing the issue and what remediation steps are necessary. Finally reference the URL to the specific section on the migration page in the documentation.

```
{
  assertions = [
    {
      assertion = config.mailserver.stateVersion != null -> config.mailserver.
↪stateVersion >= 1;
      message = ''
        Problem: The home directory for the foobar service is snafu.
        Remediation:
          - Stop the `foobar.service`
          - Rename `/var/lib/foobaz` to `/var/lib/foobar`
          - Increase the `mailserver.stateVersion` to 1.

        Check https://nixos-mailserver.readthedocs.io/en/latest/migrations.html#specific-
↪anchor-here for further details.
      '';
    }
```

(continues on next page)

(continued from previous page)

```
  ];
}
```

The setup guide should always reference the latest *stateVersion*, since we don't require any migration steps for new setups.

The migration documentation should paint a more complete picture about the steps that need to be carried out and why this has become necessary. Make sure to reference the correct anchor in the URL you put into the assertion message.

CHAPTER

FOUR

FAQ

4.1 `catchAll` users can't send email as user other than themself

To allow a `catchAll` user to send mail with the address used as recipient, the option `aliases` has to be used instead of `catchAll`.

For instance, to allow `user@example.com` to catch all mails to the domain `example.com` and send mails with any address of this domain:

```
mailserver.accounts = {
    "user@example.com" = {
        aliases = [ "@example.com" ];
    };
};
```

See also this discussion for details.

CHAPTER

FIVE

RELEASE NOTES

5.1 NixOS 26.05

5.1.1 Features

- *DKIM key management* now supports multiple selectors per domain, enabling *key rotation*. Pre-created key material is also supported. Existing automatically generated DKIM keys from before 25.11 use 1024-bit RSA and should be rotated. See *mailserver.dkim.domains*.
- Certificate handling was simplified. We recommend using the NixOS ACME module (`security.acme.certs`) and referencing a certificate configuration by name. Alternatively, certificate and private key can be managed manually. Configure either *mailserver.x509.useACMEHost* or *mailserver.x509.certificateFile* and *mailserver.x509.privateKeyFile*. See the updated *setup guide* for a basic ACME HTTP-01 example.
- Local mail accounts can now use managed cleartext passwords. This integrates well with secret management tools such as agenix and sops-nix while avoiding password leakage into the world-readable Nix store. See *mailserver.accounts.<name>.passwordFile*.
- Blocked sender responses can now be customized. This is useful if you require GDPR compliance. See *mailserver.rejectSenderMessage*.

5.1.2 Security

- TLSv1.2 cipher suites in Postfix now require AEAD and ECDHE.
- Postfix and Dovecot now support negotiation of the `SecP256r1MLKEM768` key agreement mechanism. The standardization process is ongoing.
- Deprecated and obsolete TLS signature algorithms were removed from Postfix.

5.1.3 Sieve

- **Migration**: When ManageSieve is enabled, user-created Sieve scripts must be migrated into their Dovecot home directory. See the *migration guide*.

5.1.4 LDAP

- **Migration**: Dovecot home directories for LDAP users must be migrated to UUID-based directory names. The UUID attribute can be customized through *mailserver.ldap.attributes.uuid*. See the *migration guide*.
- The LDAP configuration has been revamped. Option names have been simplified, examples and documentation improved. The *LDAP documentation* was written from the ground up.
- The default LDAP login attribute changed from `mail` to `uid`. This allows users to login with their account name rather than their email address, which is more convenient and consistent with typical LDAP practices. The exact attribute can be customized through *mailserver.ldap.attributes.username*.
- The LDAP bind password is now read verbatim without trimming whitespace. Any trailing newline is now preserved and may cause authentication failures.
- Local and LDAP accounts can now coexist. For overlapping accounts and addresses the local account will always win.

5.1.5 Internals

- Dovecot has been updated from 2.3 to 2.4 and now relies on the structured settings option.

5.1.6 Deprecations

The following integrations are deprecated and will be removed before the next release:

- `mailserver.borgbackup.enable`
- `mailserver.backup.enable`
- `mailserver.monitoring.enable`

5.2 NixOS 25.11

- The `systemName` and `systemDomain` options have been introduced to have reusable configurations for automated reports (DMARC, TLSRPT). They come with reasonable defaults, but it is suggested to check and change them as needed.
- Support for the Sender Rewriting Scheme has been added, which allows forwarding mail without breaking SPF by rewriting the envelope address.
- The default key length for new DKIM RSA keys was increased to 2048 bits as recommended in RFC 8301 3.2. We recommend rotating existing keys, as the RFC advises that signatures from 1024 bit keys should not be considered valid any longer.
- IMAP access over port `143/tcp` is now default disabled in line with RFC 8314 4.1. Use IMAP over implicit TLS on port `993/tcp` instead. If you still require this feature you can re-enable it using `mailserver.enableImap`, but it is scheduled for removal after the 25.11 release.
- SMTP server and client now support and prefer a hybrid key exchange (X25519MLKEM768)
- SMTP access over STARTTLS on port `587/tcp` is now default disabled in line with RFC 8314 3.3. If you still require this feature you can re-enable it using `mailserver.enableSubmission`.
- DMARC reports are now sent with the `noreply-dmarc` localpart from the system domain.
- DANE and MTA-STS are now validated for outgoing SMTP connections using postfix-tlspol.

- SMTP TLS connection reports (RFC 8460) are now supported using tlsrpt-reporter. They can be enabled with the `mailserver.tlsrpt.enable` option.

5.3 NixOS 25.05

- OpenDKIM has been removed and DKIM signing is now handled by Rspamd, which only supports `relaxed` canoncalizaliaton. (merge request)
- Rspamd now connects to Redis over its Unix Domain Socket by default (merge request)
  - If you need to revert TCP connections, configure `mailserver.redis.address` to reference the value of `config.services.redis.servers.rspamd.bind`.
- The integration with policyd-spf was removed and SPF handling is now fully based on Rspamd scoring. (merge request)
- Switch to the more efficient *fts-flatcurve* indexer for full text search (merge request).

  This makes use of a new index, which will be automatically re-generated the next time a folder is searched. The operation is now quick enough to be performed "just-in-time". Alternatively, all indices can be immediately re-generated for all users and folders by running

  ```
  doveadm fts rescan -u '*' && doveadm index -u '*' -q '*'
  ```

  The previous index (which is not automatically discarded to allow rollbacks) can be cleaned up by removing all the *xapian-indexes* directories within `mailserver.indexDir`.
- Individual domains can now be excluded from DMARC Reporting through `mailserver.dmarcReporting.excludedDomains`. (merge request)
- Configuring `mailserver.forwards` is now possible when the setup relies on LDAP. (merge request)
- Support for TLS 1.1 was disabled in accordance with Mozilla's recommendations. (merge request)

5.4 NixOS 24.11

- No new feature, only bug fixes and documentation improvements

5.5 NixOS 24.05

- Add new option `acmeCertificateName` which can be used to support wildcard certificates

5.6 NixOS 23.11

- Add basic support for LDAP users
- Add support for regex (PCRE) aliases

5.7 NixOS 23.05

- Existing ACME certificates can be reused without configuring NGINX
- Certificate scheme is no longer a number, but a meaningful string instead

5.8 NixOS 22.11

- Allow Rspamd to send DMARC reporting (merge request)

5.9 NixOS 22.05

- Make NixOS Mailserver options discoverable from search.nixos.org
- Add a roundcube setup guide in the documentation

5.10 NixOS 21.11

- Switch default DKIM body policy from simple to relaxed (merge request)
- Ensure locally-delivered mails have the X-Original-To header (merge request)
- NixOS Mailserver options are detailed in the documentation
- New options `dkimBodyCanonicalization` and `dkimHeaderCanonicalization`
- New option `certificateDomains` to generate certificate for additional domains (such as `imap.example.com`)

5.11 NixOS 21.05

- New *fullTextSearch* option to search in messages (based on Xapian) (Merge Request)
- Flake support (Merge Request)
- New *openFirewall* option defaulting to *true*
- We moved from Freenode to Libera Chat

5.12 NixOS 20.09

- IMAP and Submission with TLS wrapped-mode are now enabled by default on ports 993 and 465 respectively
- OpenDKIM is now sandboxed with Systemd
- New *forwards* option to forwards emails to external addresses (Merge Request)
- New *sendingFqdn* option to specify the fqdn of the machine sending email (Merge Request)
- Move the Gitlab wiki to ReadTheDocs

CHAPTER

SIX

MAILSERVER OPTIONS

6.1 mailserver

mailserver.aliases

- type: `attribute set of ((Login Account) or non-empty (list of (Login Account)))`
- default: `{ }`
- example:

```
{
  "abuse@example.com" = "user1@example.com";
  "multi@example.com" = [
    "user1@example.com"
    "user2@example.com"
  ];
  "postmaster@example.com" = "user1@example.com";
}
```

Aliases are additional mail addresses routed to one or more existing local accounts.

The target accounts are allowed to use the alias as the sender address.

Note: This feature is limited to local accounts and does not support LDAP or other external accounts.

---

mailserver.debug.all

- type: `boolean`
- default: `false`

Whether to enable verbose logging for all mailserver related services. This intended be used for development purposes only, you probably don't want to enable this unless you're hacking on nixos-mailserver.

mailserver.debug.dovecot

- type: `boolean`
- default: `config.mailserver.debug.all`

Whether to enable verbose logging for Dovecot.

mailserver.debug.rspamd

- type: boolean
- default: config.mailserver.debug.all

Whether to enable verbose logging for Rspamd.

mailserver.domains

- type: list of string
- default: [ ]
- example:

```
[
  "example.com"
]
```

The domains that this mail server serves.

mailserver.enable

- type: boolean
- default: false
- example: true

Whether to enable nixos-mailserver.

mailserver.enableImap

- type: boolean
- default: false

Whether to enable IMAP with STARTTLS on port 143.

The use of this port is deprecated per RFC 8314 4.1.

mailserver.enableImapSsl

- type: boolean
- default: true

Whether to enable IMAP with TLS in wrapper-mode on port 993.

mailserver.enableManageSieve

- type: boolean
- default: false

Whether to enable ManageSieve, setting this option to true will open port 4190 in the firewall.

The ManageSieve protocol allows users to manage their Sieve scripts on a remote server with a supported client, including Thunderbird.

mailserver.enableNixpkgsReleaseCheck

- type: boolean
- default: true

Whether to check for a release mismatch between NixOS mailserver and Nixpkgs.

Using mismatched versions is likely to cause compatibility issues and may require migrations that make an eventual rollback tricky.

It is therefore highly recommended to use a release of NixOS mailserver that corresponds with your chosen release of Nixpkgs.

mailserver.enablePop3

- type: `boolean`
- default: `false`

Whether to enable POP3 with STARTTLS on port on port 110.

The use of this port is deprecated per RFC 8314 4.1.

mailserver.enablePop3Ssl

- type: `boolean`
- default: `false`

Whether to enable POP3 with TLS in wrapper-mode on port 995.

mailserver.enableSubmission

- type: `boolean`
- default: `false`

Whether to enable SMTP with STARTTLS on port 587.

The use of this port is discouraged per RFC 8314 3.3, see also Appendix A.

mailserver.enableSubmissionSsl

- type: `boolean`
- default: `true`

Whether to enable SMTP with TLS in wrapper-mode on port 465.

mailserver.forwards

- type: `attribute set of ((list of string) or string)`
- default: `{ }`
- example:

```
{
  "gamenight@example.com" = [
    "bob@example.com"
    "frank@example.org"
    "wendy@example.net"
  ];
  "user@example.com" = "user@example.edu";
}
```

Forwards route mail from local addresses to one or more local or external addresses.

Unlike *mailserver.aliases*, the target addresses cannot send mail using the forward address.

mailserver.fqdn

- type: string
- example: "mx.example.com"

The fully qualified domain name of the mail server.

mailserver.hierarchySeparator

- type: string
- default: "."

The hierarchy separator for mailboxes used by dovecot for the namespace 'inbox'. Dovecot defaults to "." but recommends "/". This affects how mailboxes appear to mail clients and sieve scripts. For instance when using "." then in a sieve script "example.com" would refer to the mailbox "com" in the parent mailbox "example". This does not determine the way your mails are stored on disk. See https://doc.dovecot.org/2.4.4/core/config/namespaces.html#namespaces for details.

mailserver.imapMemoryLimit

- type: signed integer
- default: 256

The memory limit for the imap service, in megabytes.

mailserver.indexDir

- type: null or string
- default: null
- example: "/var/lib/dovecot/indices"

Folder to store search indices. If null, indices are stored along with email, which could not necessarily be desirable, especially when *mailserver.fullTextSearch.enable* is true since indices it creates are voluminous and do not need to be backed up.

Be careful when changing this option value since all indices would be recreated at the new location (and clients would need to resynchronize).

mailserver.lmtpMemoryLimit

- type: signed integer
- default: 256

The memory limit for the LMTP service, in megabytes.

mailserver.lmtpSaveToDetailMailbox

- type: boolean
- default: true

If an email address is delimited by a "+", should it be filed into a mailbox matching the string after the "+"? For example, user1+test@example.com would be filed into the mailbox "test".

mailserver.localDnsResolver

- type: boolean
- default: true

Runs a local DNS resolver (kresd) as recommended when running rspamd. This prevents your log file from filling up with rspamd_monitored_dns_mon entries.

mailserver.mailboxes

- type: unspecified value
- default:

```
{
  Drafts = {
    auto = "subscribe";
    special_use = "\\Drafts";
  };
  Junk = {
    auto = "subscribe";
    fts_autoindex = false;
    special_use = "\\Junk";
  };
  Sent = {
    auto = "subscribe";
    special_use = "\\Sent";
  };
  Trash = {
    auto = "no";
    fts_autoindex = false;
    special_use = "\\Trash";
  };
}
```

The default mailboxes for Dovecot maildirs.

The [special_use] option must refer to an RFC6154 2 attribute and lead with an escaped backslash.

Depending on the mail client used it might be necessary to change some mailbox's name.

mailserver.maxConnectionsPerUser

- type: signed integer
- default: 100

Maximum number of IMAP/POP3 connections allowed for a user from each IP address. E.g. a value of 50 allows for 50 IMAP and 50 POP3 connections at the same time for a single user.

mailserver.messageSizeLimit

- type: signed integer
- default: 20971520
- example: 52428800

Message size limit enforced by Postfix.

mailserver.openFirewall

- type: boolean
- default: true

Automatically open ports in the firewall.

mailserver.recipientDelimiter

- type: `string`
- default: `"+"`

Configure the recipient delimiter.

mailserver.rejectRecipients

- type: `list of string`
- default: `[ ]`
- example:

```
[
  "sales@example.com"
  "info@example.com"
]
```

Reject emails addressed to these local addresses from unauthorized senders. Use if a spammer has found email addresses in a catchall domain but you do not want to disable the catchall.

mailserver.rejectSender

- type: `list of string`
- default: `[ ]`
- example:

```
[
  "example.com"
  "spammer@example.net"
]
```

Reject emails from these addresses from unauthorized senders. Use if a spammer is using the same domain or the same sender over and over.

mailserver.rejectSenderMessage

- type: `string`
- default: `""`
- example: `"Your e-mail has not been delivered because we have blocked your e-mail address. If you believe that your e-mail address has been blocked by mistake, or if you have any other legitimate concern, please contact <address>."`

SMTP message returned to rejected senders. If not set the Postfix default will be used.

The message must be a single line and typically much shorter than 512 characters.

This could for example be used to provide a contact method (postal address, phone or alternative email) so rejected senders can exercise their Art. 21 GDPR right to object.

It is good practice to inform senders in advance that their email addresses may be processed for this purpose in accordance with Art. 13 GDPR. Storing their mail address for this purpose is generally regarded as a legitimate interest.

mailserver.rewriteMessageId

- type: boolean
- default: false

Rewrites the Message-ID's hostname-part of outgoing emails to the FQDN. Please be aware that this may cause problems with some mail clients relying on the original Message-ID.

mailserver.sendingFqdn

- type: string
- default: *mailserver.fqdn*
- example: "myserver.example.com"

The fully qualified domain name of the mail server used to identify with remote servers.

If this server's IP serves purposes other than a mail server, it may be desirable for the server to have a name other than that to which the user will connect. For example, the user might connect to mx.example.com, but the server's IP has reverse DNS that resolves to myserver.example.com; in this scenario, some mail servers may reject or penalize the message.

This setting allows the server to identify as myserver.example.com when forwarding mail, independently of *mailserver.fqdn* (which, for SSL reasons, should generally be the name to which the user connects).

Set this to the name to which the sending IP's reverse DNS resolves.

mailserver.stateVersion

- type: null or (positive integer, meaning >0)
- default: null

Tracking stateful version changes as an incrementing number.

When a new release comes out we may require manual migration steps to be completed, before the new version can be put into production.

If your stateVersion is too low one or multiple assertions may trigger to give you instructions on what migrations steps are required to continue. Increase the stateVersion as instructed by the assertion message.

mailserver.systemContact

- type: string
- example: "postmaster@example.com"

The email address where the administrative contact for this mail server is reachable.

Currently, this is only required when one of the following features is enabled:

- SMTP TLS reports (mailserver.tlsrpt.enable)

mailserver.systemDomain

- type: string
- default:

```
if config.networking.domain != null && lib.elem config.networking.domain cfg.
↪domains then
  config.networking.domain
else
  lib.head cfg.domains
```

- example: `config.networking.domain`

The primary domain used for sending automated reports.

mailserver.systemName

- type: `string`
- default: `${config.mailserver.systemDomain} mail system`
- example: `"ACME Corp."`

The sender name given in automated reports.

mailserver.useUTF8FolderNames

- type: `boolean`
- default: `false`

Store mailbox names on disk using UTF-8 instead of modified UTF-7 (mUTF-7).

mailserver.virusScanning

- type: `boolean`
- default: `false`

Whether to activate virus scanning. Note that virus scanning is *very* expensive memory wise.

6.2 mailserver.accounts

mailserver.accounts

- type: `attribute set of (submodule)`
- default: `{ }`
- example:

```
{
  user1 = {
    # This password hash leaks into the Nix store
    hashedPassword = "$y$j9T$y6eZ1o.IvVNfdGMAsUEvh1$6K/llP52uw2iDh4iSwtAn54/
↪JYy7FzCcoCHmjmx00H5";
  };
  user2 = {
    # Hashed password passed as a file
    hashedPasswordFile = "/run/keys/user2-pw-hash";
  };
  user3 = {
    # Plaintext password file
    passwordFile = "/run/keys/user3-pw";
  };
}
```

Attribute set of mail accounts.

Each entry defines a mailbox and login credentials, where the attribute name is used as the login username and optionally routed mail address.

Use `mkpasswd` to generate password hashes.

`mailserver.accounts.<name>.aliases`

- type: `list of string`
- default: `[ ]`
- example:

```
[
  "abuse@example.com"
  "postmaster@example.com"
]
```

List of additional mail addresses (aliases) that get routed to this account.

Catch-all with sending permissions

Configure `@example.com` to create a catch-all for this domain that also allows sending from all addresses.

`mailserver.accounts.<name>.aliasesRegexp`

- type: `list of string`
- default: `[ ]`
- example:

```
[
  "/^tom\\..*@domain\\.com$/"
]
```

Same as *`mailserver.accounts.<name>.aliases`* but using PCRE (Perl compatible regex).

`mailserver.accounts.<name>.catchAll`

- type: `list of value "example.com" (singular enum)`
- default: `[ ]`
- example:

```
[
  "example.com"
  "example2.com"
]
```

For which domains should this account act as a catch all?

Warning: Does not allow sending from all addresses of these domains. Use *`mailserver.accounts.<name>.aliases`* if that is required.

`mailserver.accounts.<name>.hashedPassword`

- type: `null or string`
- default: `null`

- example: `"$y$j9T$vfGrwkAaXCjCEWtVNMQck1$383uIXQmn2z0hnmVAA8kwFQmjNj78.nYbvWeyNLIaP1"`

The hashed login password for this account.

Use `mkpasswd` to create password hashes:

```
nix-shell -p mkpasswd --run 'mkpasswd -s'
```

Note: This is a convenience option, when your threat model allows storing hashed secrets in the world-readable Nix store.

Passing the hash through *mailserver.accounts.<name>.hashedPasswordFile* allows relying on filesystem discretionary access control as another security boundary.

---

mailserver.accounts.<name>.hashedPasswordFile

- type: `null or absolute path`
- default: `null`
- example: `"/run/keys/user1-pw-hash"`

The hashed login password for this account read from a file.

Use `mkpasswd to create password hashes:

```
nix-shell -p mkpasswd --run 'mkpasswd -s'
```

mailserver.accounts.<name>.passwordFile

- type: `null or path not in the Nix store`
- default: `null`
- example: `"/run/keys/user1-pw"`

The plaintext login password for this account read from a file.

Note: The password is hashed before it is passed on to Dovecot.

---

mailserver.accounts.<name>.quota

- type: `null or string`
- default: `null`
- example: `"2G"`

The quota limit for this user.

The value is must use a size format like `500M`, `2G`, `10G`.

If unset, will fall back to *mailserver.quota.defaults.perUser* if set.

mailserver.accounts.<name>.sendOnly

- type: `boolean`
- default: `false`

Specifies if the account should be a send-only account.

Emails sent to send-only accounts will be rejected with the reason configured in *mailserver.accounts.<name>.sendOnlyRejectMessage*.

mailserver.accounts.<name>.sendOnlyRejectMessage

- type: `string`
- default: `"This account cannot receive emails."`

The message returned to the sender for a send-only account.

See *mailserver.accounts.<name>.sendOnly*.

mailserver.accounts.<name>.sieveScript

- type: `null or strings concatenated with "\n"`
- default: `null`
- example:

```
''
  require ["fileinto", "mailbox"];

  if address :is "from" "gitlab@mg.gitlab.com" {
    fileinto :create "GitLab";
    stop;
  }

  # This must be the last rule, it will check if list-id is set, and
  # file the message into the Lists folder for further investigation
  elsif header :matches "list-id" "<?*>" {
    fileinto :create "Lists";
    stop;
  }
''
```

Per-user sieve script.

6.3 mailserver.x509

mailserver.x509.certificateFile

- type: `null or absolute path`
- default: `null`
- example: `"/var/keys/certs/fullchain.pem"`

Path to the signed X509 certificate including intermediate certificates.

This is commonly referred to as `fullchain.pem`.

Mutually exclusive with *mailserver.x509.useACMEHost*.

mailserver.x509.privateKeyFile

- type: `null or string`

- default: `null`
- example: `"/var/keys/certs/privkey.pem"`

Path to the X509 private key.

This is commonly referred to as `privkey.pem`.

Mutually exclusive with *mailserver.x509.useACMEHost*.

mailserver.x509.useACMEHost

- type: `null or string`
- default: `null`
- example: `config.mailserver.fqdn`

Common name used in the relevant `security.acme.certs` configuration.

Mutually exclusive with *mailserver.x509.certificateFile* and *mailserver.x509.privateKeyFile*.

6.4 mailserver.storage

mailserver.storage.directoryLayout

- type: `one of "fs", "Maildir++"`
- default: `"Maildir++"`

Sets whether dovecot should organize mail in subdirectories:

- /var/vmail/example.com/user/.folder.subfolder/ (Maildir++ layout)
- /var/vmail/example.com/user/folder/subfolder/ (FS layout)

See https://doc.dovecot.org/2.4.4/core/config/mailbox_formats/maildir.html#directory-layout for further details.

mailserver.storage.gid

- type: `positive integer, meaning >0`
- default: `5000`

The group id of the primary group of the vmail user.

This group owns the mail storage directories. Access can be delegated to other users via group membership.

> **Warning:** If you change this value you also need to manually adjust the ownership of your *mailserver.storage.path*.

mailserver.storage.group

- type: `string`
- default: `"virtualMail"`

The primary group name of the user that owns the *mailserver.storage.path*.

mailserver.storage.owner

- type: `string`
- default: `"virtualMail"`

The name of the user that owns the *mailserver.storage.path*.

mailserver.storage.path

- type: `absolute path`
- default: `"/var/vmail"`

Path on disk where mail home directories are stored.

mailserver.storage.uid

- type: `positive integer, meaning >0`
- default: `5000`

The user id assigned to the vmail user.

This user owns the mail storage files and directories and is used by services accessing the mail store.

> **Warning:** If you change this value you also need to manually adjust the ownership of your *mailserver.storage.path*.

6.5 mailserver.dkim

mailserver.dkim.defaults.keyLength

- type: `signed integer`
- default: `2048`

The default key length used for generating new DKIM keys.

Only applies for RSA keys, Ed25519 keys use a fixed key length.

Per RFC8301 3.2 the minimum RSA key length should be at least 2048 bit.

This value should most likely not be changed. Once DKIM keys for domain and selector are generated changing this value will not regenerate the keypair. Instead create a new selector and configure *mailserver.dkim.domains.<name>.selectors.<name>.keyLength*.

mailserver.dkim.defaults.keyType

- type: `one of "rsa", "ed25519"`
- default: `"rsa"`

The key type used for generating DKIM keys. Ed25519 support was introduced in RFC6376 (2018).

> **Warning:** Ed25519 DKIM keys are currently not recommended for sole use, as various DKIM validators out there lack support and consider the keypair invalid.

This value should most likely not be changed. Once DKIM keys for domain and selector are generated changing this value will not regenerate the keypair. Instead create a new selector and configure *mailserver.dkim.domains.<name>.selectors.<name>.keyType*.

mailserver.dkim.defaults.selector

- type: `string`
- default: `"mail"`

The default selector used to reference and lookup DKIM keys.

This value should most likely not be changed. Instead manage *mailserver.dkim.domains.<name>.selectors* to sign with one or multiple DKIM key pairs and manage migrations.

mailserver.dkim.domains

- type: `attribute set of (submodule)`
- default: `{ }`
- example:

```
{
  "example.com".selectors = {
    "mail" = {
      # inherit defaults from mailserver.dkim.defaults
    };
    "rsa-2026-03".keyFile = "/run/keys/example.com-dkim-rsa-2026-03.key";
  };
};
```

DKIM configuration per domain.

mailserver.dkim.domains.<name>.selectors

- type: `attribute set of (submodule)`
- default: `{ }`
- example:

```
{
  "mail" = {
    # inherit defaults from mailserver.dkim.defaults
  };
  "rsa-2026-03".keyFile = "/run/keys/example.com-dkim-rsa-2026-03.key";
};
```

DKIM selectors used for signing outgoing mail for this domain.

When no selector is configured a default selector will be created with settings inherited from *mailserver.dkim.defaults*.

mailserver.dkim.domains.<name>.selectors.<name>.keyFile

- type: `null or path not in the Nix store`
- default: `null`
- example: `"/run/keys/example.com-dkim-rsa-2026-03.key"`

Path to an existing DKIM private key file.

DKIM keys can be generated using `rspamadm dkim_keygen`.

This option is mutually exclusive with `keyType` and `keyLength`.

mailserver.dkim.domains.<name>.selectors.<name>.keyLength

- type: `null or signed integer`
- default: `null`
- example: `2048`

The key length used for generating this DKIM key.

Only applies for RSA keys, Ed25519 keys use a fixed key size.

This option is mutually exclusive with `keyFile`.

mailserver.dkim.domains.<name>.selectors.<name>.keyType

- type: `null or one of "rsa", "ed25519"`
- default: `null`
- example: `"rsa"`

The key type used for generating this DKIM keypair.

> **Warning:** Ed25519 DKIM keys are currently not recommended for sole use, as various DKIM validators out there lack support and consider the keypair invalid.

This option is mutually exclusive with `keyFile`.

mailserver.dkim.enable

- type: `boolean`
- default: `true`
- example: `true`

Whether to enable DKIM signing.

mailserver.dkim.keyDirectory

- type: `absolute path`
- default: `"/var/dkim"`

The path where DKIM siging keys are stored.

6.6 mailserver.srs

mailserver.srs.domain

- type: null or string
- default: config.mailserver.systemDomain
- example: "srs.example.com"

Mail domain used for ephemeral SRS envelope addresses.

Note: This domain can only support relaxed SPF alignment.

Important: For privacy reasons you should use a dedicated domain when serving multiple unrelated domains.

mailserver.srs.enable

- type: boolean
- default: false
- example: true

Whether to enable Sender Rewrite Scheme.

6.7 mailserver.dmarcReporting

mailserver.dmarcReporting.enable

- type: boolean
- default: false

Whether to send out aggregated, daily DMARC reports in response to incoming mail, when the sender domain defines a DMARC policy including the RUA tag.

This is helpful for the mail ecosystem, because it allows third parties to get notified about SPF/DKIM violations originating from their sender domains.

See https://rspamd.com/doc/modules/dmarc.html#reporting

mailserver.dmarcReporting.excludeDomains

- type: list of string
- default: [ ]

List of domains or eSLDs to be excluded from DMARC reports.

6.8 mailserver.tlsrpt

mailserver.tlsrpt.enable

- type: boolean
- default: false
- example: true

Whether to enable delivery of SMTP TLS reports according to RFC 8460.

6.9 mailserver.fullTextSearch

mailserver.fullTextSearch.autoIndex

- type: boolean
- default: true

Enable automatic indexing of messages as they are received or modified.

Tip: Can be overridden per mailbox by setting fts_autoindex for *mailserver.mailboxes*. By default the Junk and Trash folders are already excluded.

---

mailserver.fullTextSearch.enable

- type: boolean
- default: false
- example: true

Whether to enable Full text search indexing with Xapian through the fts_flatcurve plugin. This has significant performance and disk space cost. .

mailserver.fullTextSearch.fallback

- type: boolean
- default: true

Whether to fallback to slow non-indexed search, if FTS lookup and indexing have failed.

See https://doc.dovecot.org/2.4.4/core/plugins/fts.html#fts_search_read_fallback.

mailserver.fullTextSearch.filters

- type: list of string
- default:

```
[
  "normalizer-icu"
  "snowball"
  "stopwords"
]
```

The list of language filters to apply.

mailserver.fullTextSearch.headerExcludes

- type: list of string
- default:

```
[
  "Received"
  "DKIM-*"
  "X-*"
  "Comments"
]
```

The list of headers to exclude while indexing.

See https://doc.dovecot.org/2.4.4/core/plugins/fts.html#fts_header_excludes.

mailserver.fullTextSearch.languages

- type: non-empty (list of string)
- default:

```
[
  "en"
]
```

- example:

```
[
  "en"
  "de"
]
```

A list of languages that the full text search should detect.

At least one language must be specified. The language listed first is the default and is used when language recognition fails.

See https://doc.dovecot.org/2.4.4/core/plugins/fts.html#languages.

mailserver.fullTextSearch.memoryLimit

- type: null or signed integer
- default: null
- example: 1024

Memory limit for the indexer process, in MiB.

When null the [default_vsz_limit] applies while with 0 no limit is applied.

default_vsz_limit

mailserver.fullTextSearch.substringSearch

- type: boolean
- default: false

Whether to allows substring searches. By default only prefix searches are supported.

> **Warning:** Enabling this significantly increases storage requirements.

See https://doc.dovecot.org/2.4.4/core/plugins/fts_flatcurve.html#fts_flatcurve_substring_search.

6.10 mailserver.quota

mailserver.quota.defaults.perUser

- type: `null or string`
- default: `null`
- example: `"10G"`

Default quota applied to all users.

The value must use a size format like `500M`, `2G`, `10G`.

If set to `null`, no default per user quota is applied and only explicit per user quotas apply, if set.

mailserver.quota.enable

- type: `boolean`
- default: `true`

Whether to enable quota support.

When enabled, incoming mail can be rejected if a mailbox exceeds its quota.

mailserver.quotaStatusMemoryLimit

- type: `signed integer`
- default: `256`

The memory limit for the quota-status service, in megabytes.

6.11 mailserver.redis

mailserver.redis.address

- type: `string`
- default: `config.services.redis.servers.rspamd.unixSocket`

Path, IP address or hostname that Rspamd should use to contact Redis.

mailserver.redis.configureLocally

- type: `boolean`
- default: `true`

Whether to provision a local Redis instance.

mailserver.redis.password

- type: null or string
- default: config.services.redis.servers.rspamd.requirePass

Password that rspamd should use to contact redis, or null if not required.

mailserver.redis.port

- type: null or 16 bit unsigned integer; between 0 and 65535 (both inclusive)
- default: null
- example: config.services.redis.servers.rspamd.port

Port that Rspamd should use to contact Redis.

6.12 mailserver.ldap

mailserver.ldap.attributes.mail

- type: string
- default: "mail"
- example: "maildrop"

The attribute name used for looking up accounts by mail address.

Typically this can be the mail attribute from the inetOrgPerson schema, or the maildrop attribute from the unofficial Postfix schema.

mailserver.ldap.attributes.password

- type: null or string
- default: null
- example: "userPassword"

The LDAP attribute referencing the account password used to login with.

The account passwords stored in LDAP must be hashed with a supported Password Scheme in order for Dovecot to understand them.

Typically the userPassword attribute which is part of the inetOrgPerson schema.

If null, Authentication Binds will be used instead.

mailserver.ldap.attributes.username

- type: string
- default: "uid"
- example: "name"

The LDAP attribute referencing the username used to login with.

Typically the uid attribute which is part of the inetOrgPerson schema.

mailserver.ldap.attributes.uuid

- type: string
- default: "entryUUID"
- example: "uuid"

The long-term stable LDAP attribute to reference accounts across username changes. Used to determine a stable Dovecot home and mail directory location.

Typically the entryUUID attribute as defined by RFC4530.

mailserver.ldap.base

- type: string
- example: "ou=people,ou=accounts,dc=example,dc=com"

Base DN below which user accounts are searched for.

mailserver.ldap.bind.dn

- type: string
- example: "cn=mail,ou=accounts,dc=example,dc=com"

DN used to bind against the LDAP server.

The server uses this account to lookup and filter accounts.

mailserver.ldap.bind.passwordFile

- type: path not in the Nix store
- example: "/run/my-secret"

File containing the password required to bind against the LDAP server.

> **Warning:** The password file is read verbatim. Any trailing newline will become part of the password and may cause authentication failures.

mailserver.ldap.caFile

- type: absolute path
- default: ${pkgs.cacert}/etc/ssl/certs/ca-bundle.crt

Bundle of CA certificates used to authenticate the LDAP server certificate.

mailserver.ldap.dovecot.passFilter

- type: null or string
- default:

```
with config.mailserver.ldap.attributes; "${username}=%{user}";
```

- example: "(&(memberOf=cn=mail_users,ou=groups,dc=example,dc=com)(uid=%{user}))"

LDAP filter used to restrict which users are eligible to authenticate against Dovecot.

See the passdb_ldap_filter reference in the Dovecot manual.

mailserver.ldap.dovecot.userFilter

- type: string
- default:

```
with config.mailserver.ldap.attributes; "(|(${mail}=%{user})(${username}=%{user}
↪))";
```

- example: "(|(mail=%{user})(uid=%{user}))"

LDAP filter used for LMTP delivery from Postfix and post-login information construction, like the home directory.

See the userdb_ldap_filter reference at in the Dovecot manual.

mailserver.ldap.enable

- type: boolean
- default: false
- example: true

Whether to enable LDAP support.

mailserver.ldap.postfix.filter

- type: string
- default:

```
with config.mailserver.ldap.attributes; "${mail}=%s";
```

- example: "(mail=%s)"

LDAP filter used to search for an account by mail, where %s is a substitute for the address in question.

mailserver.ldap.scope

- type: one of "base", "one", "sub"
- default: "sub"

Search scope relative to the *mailserver.ldap.base*.

- base: Only the exact Base DN
- one: Immediate child entries of the Base DN, but not the Base DN itself.
- sub: Base DN and all descendant entries at any depth.

In practice only one or sub are suitable for multiple LDAP users.

mailserver.ldap.startTls

- type: boolean
- default: false

Whether to enable StartTLS on ldap:// connections.

mailserver.ldap.uris

- type: list of string
- default: [ ]

- example:

```
[
  "ldaps://ldap1.example.com"
  "ldaps://ldap2.example.com"
]
```

List of LDAP server URIs. Multiple can be specified.

Use `ldaps://` for implicit TLS or `ldap://` for a plain connection. See also *mailserver.ldap.startTls* to enable StartTLS on plain connections.

6.13 mailserver.workarounds

mailserver.workarounds.all

- type: `boolean`
- default: `true`

Whether to enable various workarounds for compatibility with other mail servers or clients. This is safe to enable and workarounds come and go. At times, this option may have no effect at all.

mailserver.workarounds.digicertRootDiscontinuation

- type: `boolean`
- default: `config.mailserver.workarounds.all`

Whether to patch Postfix's CA bundle to continue trusting the discontinued DigiCert Root CA.

This will allow Postfix to deliver mails to mail servers that still serve a certificate signed by this CA. Without this patch connections that enforce DANE or MTA-STS will break.

Microsoft Outlook is known to do this.

CHAPTER

SEVEN

MIGRATIONS

With mail server configuration best practices changing over time we might need to make changes that require you to complete manual migration steps before you can deploy a new version of NixOS mailserver.

The initial *mailserver.stateVersion* value should be copied from the setup guide that you used to initially set up your mail server. If in doubt you can always initialize it at 1 and walk through all assertions, that might apply to your setup.

7.1 NixOS 26.05

7.1.1 #5 Sieve script directory migration

Sieve scripts managed by users via ManageSieve were previously stored in `/var/sieve` (or via the now-removed option `mailserver.sieveDirectory`). This setup partially mirrored the mail directory structure in `/var/vmail` (`mailserver.storage.path`), which proved to be fragile and error-prone.

Thanks to a *prior migration*, we can now migrate these directories into each user's home directory, aligning with upstream recommendations — almost nine years later.

This migration is only required if you have *mailserver.enableManageSieve* enabled.

1. If you are coming from `25.11` and are using LDAP, temporarily disable *mailserver.enableManageSieve* by setting it to `false`, deploy, and only then continue with this migration.

   If you are not coming from `25.11` or are not using LDAP, continue with this migration as is.

2. Copy the migration script to your mailserver and make it executable:

   ```
   cd /tmp
   wcurl https://gitlab.com/simple-nixos-mailserver/nixos-mailserver/-/raw/main/
   ↪migrations/nixos-mailserver-migration-05.py
   chmod +x nixos-mailserver-migration-05.py
   ```

3. Stop the `postfix.service`.

   ```
   systemctl stop postfix.service
   ```

4. Create a backup or snapshot of your `mailserver.sieveDirectory`, so you can restore should anything go wrong.

5. Run the migration script and pass your `mailserver.sieveDirectory` as argument:

   The script should be run under the user who owns the `mailserver.sieveDirectory`. If run as root it will automatically switch into the appropriate user by itself.

The script will not modify your data unless called with `--execute`.

The migration script finds all Sieve script directories in `/var/sieve/` (or any other `mailserver.sieveDirectory`), for example that of `bob` at `/var/vmail/bob@example.com`.

It then takes `bob@example.com` and looks up the home directory for `bob`. Finally, it starts suggesting the necessary move operations to migrate the Sieve directory to `/var/vmail/example.com/bob/sieve` and symlinks the active script to `/var/vmail/example.com/bob/.dovecot.sieve`.

Example:

```
./nixos-mailserver-migration-05.py \
   /var/sieve
```

6. Review the script output.

   The script can highlight various inconsistencies and problems, that should be reviewed and acted upon.

   If in doubt, join our community chat for help before applying any changes.

7. Rerun the command with `--execute` or run the proposed commands manually.
8. Update the `mailserver.stateVersion` to `5`.
9. The previous Sieve directory (`mailserver.sieveDirectory`) should now be safe to delete.
10. If you temporarily disabled *mailserver.enableManageSieve* in step 1, re-enable it now by setting it back to `true`.

7.1.2 #4 Dovecot LDAP UUID-based home directories

LDAP Support in NixOS mailserver was introduced during the 23.11 release cycle and came with a number of flaws that we are correcting now, three years later.

This particular migration is needed because up until now we were relying on email addresses to construct the Dovecot home directory path (`var/vmail/ldap/user@example.com`) which is fragile: addresses can change, requiring manual homedir relocation. Switching to UUID-based homedirs (`/var/vmail/ldap/<uuid>`) ensures stable, unique paths and applies well-known best practices to mailserver management.

This migration is only required if you have *mailserver.ldap.enable* enabled.

1. Copy the migration script to your mailserver and make it executable:

```
cd /tmp
wcurl https://gitlab.com/simple-nixos-mailserver/nixos-mailserver/-/raw/main/
↪migrations/nixos-mailserver-migration-04.py
chmod +x nixos-mailserver-migration-04.py
```

2. Stop the `dovecot.service`.

```
systemctl stop dovecot.service
```

3. Create a backup or snapshot of your *mailserver.storage.path*, so you can restore should anything go wrong.
4. Run the migration script and pass the required arguments to enable LDAP lookups:

   The script should be run under the user who owns the *mailserver.storage.path*. If run as root it will automatically switch into the appropriate user by itself.

   The script will not modify your data unless called with `--execute`.

The migration script finds all Dovecot home directories in `/var/vmail/ldap/` (or any other *mailserver.storage.path*), for example that of bob at `/var/vmail/ldap/bob@example.com`. It then takes `bob@example.com` and queries the LDAP server for `mail=bob@example.com` to retrieve the UUID attribute. Finally it starts suggesting the necessary move operations to arrive at `/var/vmail/ldap/f3b4e8ea-087f-42cc-95f0-cbfd99386092` for bob.

Example:

```
./nixos-mailserver-migration-04.py \
  --ldap-uri ldaps://ldap1.example.com
  --ldap-bind-dn cn=mail,ou=accounts,dc=example,dc=com \
  --ldap-bind-pw-file /run/keys/ldap-bind-pw \
  --ldap-base ou=people,ou=accounts,dc=example,dc=com \
  --ldap-scope sub \
  --ldap-filter "(mail=%s)" \
  --ldap-attr-uuid entryUUID \
   /var/vmail
```

For the `--ldap-attr-uuid` parameter we expect a long-term stable identifier, ideally a UUID field. The exact attribute name depends on your LDAP implementation, for example:

- Authentik: `uid` [1]
- Kanidm: `uuid` [2]
- Keycloak `entryUUID`
- OpenLDAP: `entryUUID` (RFC4530)

If your LDAP provider isn't listed you can determine the correct attribute by querying a user entry with `ldapsearch`. Finally, configure *mailserver.ldap.attributes.uuid* accordingly.

Add `--ldap-starttls` if you use the the *ldap://* URI scheme and require explicit TLS.

5. Review the script output.

   It's primary job is to determine the UUID for an LDAP account, so that it can rename the Dovecot home directory from mail address to UUID within the same directory.

   The script can highlight various inconsistencies and problems, that should be reviewed and acted upon.

   If in doubt, join our community chat for help before applying any changes.

6. Rerun the command with `--execute` or run the proposed commands manually.
7. Update the `mailserver.stateVersion` to `4`.

7.2 NixOS 25.11

7.2.1 #3 Dovecot mail directory migration

The way the Dovecot home directory for login accounts were previously set up resulted in shared home directories for all those users. This is not a supported Dovecot configuration.

To resolve this we migrated the home directory into the individual *domain/localpart* subdirectory below the *mailserver.mailDirectory*.

But since this now overlaps with the location of the Maildir, it must be migrated into the *mail/* directory below the home directory. And while the LDAP home directory is not affected we use this migration to keep the Maildir configurations of LDAP users in sync with those of local accounts.

This is a big step forward, since we can now more cleanly colocate other data directories, like sieve in the home directory, which in turn simplifies backups.

This migration is required for every configuration.

For remediating this issue the following steps are required:

1. Copy the migration script script to your mailserver and make it executable:

```
wcurl https://gitlab.com/simple-nixos-mailserver/nixos-mailserver/-/raw/main/
↪migrations/nixos-mailserver-migration-03.py
chmod +x nixos-mailserver-migration-03.py
```

2. Stop the `dovecot.service`.

```
systemctl stop dovecot.service
```

3. Create a backup or snapshot of your `mailserver.mailDirectory`, so you can restore should anything go wrong.
4. Run the migration script under your virtual mail user with the following arguments:
   - `--layout default` unless `useFSLayout` is enabled, then `--layout folder`
   - The value of `mailserver.mailDirectory`, which defaults to `/var/vmail`

   The script should be run under the user who owns the `mailDirectory`. If run as root it will try to switch into the appropriate user by itself.

   The script will not modify your data unless called with `--execute`.

   Example:

```
./nixos-mailserver-migration-03.py --layout default /var/vmail
```

5. Review the commands. They should be
   - create a `mail` directory for each account,
   - move maildir contents from the parent directory into it,
   - suggest removal of files that do not belong to the maildir
     - their removal is not mandatory and the script **will not** remove them when called with `--execute`
     - review these items carefully if you want to remove them yourself
   - remove obsolete files from the old home directory location
6. Rerun the command with `--execute` or run the commands manually.
7. Update the `mailserver.stateVersion` to `3`.

7.2.2 #2 Dovecot LDAP home directory migration

The Dovecot configuration for LDAP home directories previously did not respect the `mailserver.mailDirectory` setting.

This means that home directories were unconditionally located at `/var/vmail/ldap/%{user}`.

This migration is required if you both:

- enabled the LDAP integration (`mailserver.ldap.enable`)
- and customized the default mail directory (`mailserver.mailDirectory != "/var/vmail"`)

For remediating this issue the following steps are required:

1. Stop `dovecot.service`.
2. Move `/var/vmail/ldap` below your `mailserver.mailDirectory`.
3. Update the `mailserver.stateVersion` to `2`.

7.2.3 #1 Initialization

This option was introduced in the NixOS 25.11 release cycle, in which case you can safely initialize its value at *1*.

```
mailserver.stateVersion = 1;
```

CHAPTER

EIGHT

LDAP

LDAP (Lightweight Directory Access Protocol) is a protocol for accessing and managing a centralized directory of user and group information. It can be used to authenticate users and provide a single source of truth for email accounts and aliases across mail services.

8.1 Requirements

To enable the LDAP integration the following requirements must be fulfilled:

- Existing LDAP service (we currently only test against OpenLDAP)
- Bind credentials against LDAP with permissions to
  - search for the acceptable set of users
  - read the *mailserver.ldap.attributes.password* attribute
- Each user entry must provide attributes that can serve as
  - *mailserver.ldap.attributes.mail* (primary mail address)
  - *mailserver.ldap.attributes.username* (login name)
  - *mailserver.ldap.attributes.password* (login password)
  - *mailserver.ldap.attributes.uuid* (stable identifier)

8.2 Features

We currently have a basic feature set covering user accounts only and try to follow best practices to simplify maintenance.

- Users authenticate with the username and password attribute
- Maildir storage paths are constructed using the uuid attribute
- Primary mail address read from mail attribute

8.3 Limitations

8.3.1 Design choices

These are intentional choices in how the mail server operates that affect the LDAP integration.

- For mail address routing local accounts always take priority over LDAP accounts.

8.3.2 Planned

These are features we are interested in but require implementation, documentation and tests.

- Aliases based on LDAP attributes
- Quotas based on LDAP attributes

8.3.3 Avoided

The following features will likely never be implemented, since they would complicate the setup significantly.

- Domains based on LDAP entries (would require integration with everything we already do for *mailserver.domains*)
- Use of `homeDirectory`, `uid`, `gid` LDAP attributes (we are committed to a virtual setup with one vmail user/uid/gid and UUID based home directories)
- Declarative aliases through *mailserver.aliases*. These are limited to local accounts, because Postfix enforces sender ownership based on login identity and does not consult virtual aliases for authorization.

8.4 Enabling LDAP support

Enable the LDAP integration by configuring an authenticated LDAP connection and how to locate all users. The bind DN must be allowed to read the configured password attribute, which may require additional configuration

```
{
  mailserver = {
    ldap = {
      enable = true;
      uris = [
        "ldaps://ldap1.example.com"
        "ldaps://ldap2.example.com"
      ];
      bind = {
        dn = "cn=mail,dc=example=dc=com";
        passwordFile = "/run/keys/ldap-bind-pw";
      };
      base = "ou=users,dc=example,dc=com";
      scope = "one";
    };
  };
}
```

We provide sensible defaults for each attribute, that can be adapted to your local setup.

```
{
  mailserver = {
    ldap = {
      attributes = {
        uuid = "entryUUID";
        username = "uid";
        password = "userPassword";
        mail = "mail";
      };
    };
  };
}
```

Refer to our LDAP test for an complete example, and see the LDAP options section for all possible settings.

CHAPTER

NINE

DKIM SIGNING

DKIM (DomainKeys Identified Mail) is an email authentication mechanism that allows a mailserver to digitally sign outgoing emails for a domain. Receiving mail servers can verify this signature using a public key published in DNS to confirm the message was authorized by the domain and was not modified during transit.

9.1 How DKIM works in practice

1. `bob@bar.example` sends an email to `alice@foo.example`. The sending mail server for `bar.example` selects one or multiple DKIM keys using a **selector** (e.g., `mail`) and creates one or multiple cryptographic signature for selected headers and the message body, adding a `DKIM-Signature` header that references the selector.

2. The message arrives at `foo.example`. The receiving mail server reads the `DKIM-Signature` headers, looks up the public keys for `bar.example` for each of the specified selectors (e.g., `mail._domainkey.bar.example`), and verifies that at least one signature matches the message content.

3. With a valid signature, the receiver knows the message was authorized by `bar.example` and that the signed headers and body were not modified in transit. If the content or signed headers were changed, the DKIM verification fails. The use of selectors allows `bar.example` to rotate or migrate keys without disrupting verification for previously sent messages.

9.2 Enabling DKIM Signing

Because DKIM signing is crucial for reliable mail delivery it is enabled by default and without further configuration a DKIM keypair will be generated for each *`mailserver.domains`* (including *`mailserver.srs.domain`*, if set) based on *`mailserver.dkim.defaults`*.

```
{
  mailserver = {
    domains = [ "example.com" ];
    dkim.enable = true; # enabled by default
  };
}
```

Note: If you set up NixOS Mailserver before the 25.11 release your DKIM keys were generated with 1024 bit RSA and we recommend replacing them with 2048 bit RSA key material per RFC8301 3.2.

9.3 DKIM Key Rotation

DKIM key rotation replaces a domain's signing keys to maintain strong email authentication and support algorithm upgrades. Rotation is essential for migrating away from weaker or deprecated algorithms.

Selectors allow multiple keys to coexist during the transition: a new key can be deployed under a different selector while the old key remains valid for a limited period to verify messages still in transit. Once all messages signed with the old key have been delivered, the key can be safely retired, ensuring a reliable migration without breaking verification.

9.3.1 1. Make the automatically generated key explicit

First we need to make sure we keep the current DKIM key configured. If you were relying on automatically generated keys before, you now need to start explicitly defining that key, because explicit selector configuration takes precedence.

```
{
  mailserver = {
    domains = [ "example.com" ];
    dkim.domains = {
      "example.com".selectors = {
        "${config.mailserver.dkim.defaults.selector}" = { };
      };
    };
  };
}
```

9.3.2 2. Create the new DKIM keypair

Next we need to create a new DKIM key with a unique selector, you can for example choose the current date. Without any settings passed a new key will be generated from the current *mailserver.dkim.defaults*, which should be sufficient.

```
{
  mailserver = {
    domains = [ "example.com" ];
    dkim = {
      enable = true;
      domains."example.com".selectors = {
        "${config.mailserver.dkim.defaults.selector}" = { };
        "rsa-2026-03" = {
          keyType = "rsa";
          keyLength = 2048;
        };
      };
    };
  };
}
```

Warning: While DKIM does support Ed25519 keys (RFC8463), many validators still lack proper support and may treat Ed25519 key material as invalid. As a result, mail signed only with Ed25519 DKIM keys may fail verification at some receivers.

Once this configuration is applied the new keypair will be generated below *mailserver.dkim.keyDirectory*, which defaults to `/var/dkim`. The mailserver then begins signing outgoing mail with this key, so that it is now signing with two DKIM keys simultaneously.

To allow receiving servers to verify the new DKIM signature its public key needs to be published into DNS. Look up the public key from `/var/dkim/example.com.rsa-2026-03.txt` and create the following DNS record by substituting selector and public key.

Name	TTL	Type	Value
rsa-2026-03._domainkey.example.com.	86400	TXT	v=DKIM1; k=rsa; p=<public key>

Note: If you created an Ed25519 key, make sure to set the correct key type: `k=ed25519`

Now wait for a few minutes and then check DNS propagation to show the value specified.

```
$ nix-shell -p dig --command "dig @ns1.example.org TXT rsa-2026-03._domainkey.example.com
↪"
```

You can use https://www.mail-tester.com to test the new DKIM signature passes validation. They allow you to view the message source where you can check that the correct number of `DKIM-Signature` keys are present in the mail header.

9.3.3 3. Stop signing with the old DKIM keypair

Once validation passes we need to stop signing with the old DKIM keypair, so mail in transit eventually stops depending on the key material we want to rotate out. Removing the selector will not remove the key material from disk, but it will stop using it to sign outgoing mail.

```
{
  mailserver = {
    domains = [ "example.com" ];
    dkim = {
      enable = true;
      domains."example.com".selectors = {
        "rsa-2026-03" = {
          keyType = "rsa";
          keyLength = 2048;
        };
      };
    }
  }
}
```

Apply the configuration.

9.3.4 4. Remove the old DKIM selector from DNS

> **Warning:** Do not remove the DNS records for the old selector immediately. Keeping them in place is essential to ensure that messages still in transit can be verified and delivered successfully.

Mail delivery is not always instantaneous. In the worst case, multiple retries over several days may be required. According to RFC5321 4.5.4.1 delivery should be retried for at least 4-5 days.

This means messages signed only with the old DKIM key could still be in transit and rely on the old selector to verify their signatures. To ensure reliable delivery, we recommend waiting **at least five days** before removing the old DKIM selector from DNS.

CHAPTER

TEN

FULL TEXT SEARCH

By default, when your IMAP client searches for an email containing some text in its *body*, dovecot will read all your email sequentially. This is very slow and IO intensive. To speed body searches up, it is possible to *index* emails with the `fts_flatcurve` dovecot plugin.

10.1 Enabling full text search

To enable indexing for full text search here is an example configuration.

```
{
  mailserver = {
    # ...
    fullTextSearch = {
      enable = true;
      # index new email as they arrive
      autoIndex = true;
      # only query index
      fallback = false;
    };
  };
}
```

Disabling the *mailserver.fullTextSearch.fallback* option tells dovecot to fail any body search query that cannot use an index. This prevents Dovecot to fall back to the IO-intensive brute force search.

If you set *mailserver.fullTextSearch.autoIndex* to `false`, indices will be created when the IMAP client issues a search query, so latency will be high.

10.2 Resource requirements

Indices created by the full text search feature can take more disk space than the emails themselves. By default, they are kept within the maildir. When enabling the full text search feature, it is recommended to move indices in a different location, such as (`/var/lib/dovecot/indices`) by configuring *mailserver.indexDir*.

Warning: When the value of the *mailserver.indexDir* option is changed, all dovecot indices needs to be recreated: clients would need to resynchronize.

Indexation itself is rather resource intensive, in CPU, and for emails with large headers, in memory as well. Initial indexation of existing emails can take hours. If the indexer worker is killed or segfaults during indexation, it can be that it tried to allocate more memory than allowed. You can increase the default memory limit through *mailserver.fullTextSearch.memoryLimit*.

10.3 Mitigating resources requirements

You can:

- exclude some headers from indexation with *mailserver.fullTextSearch.headerExcludes*
- disable expensive token normalisation in *mailserver.fullTextSearch.filters*
- disable automatic indexation for individual mailboxes by overriding fts_autoindex on the mailbox level. This is exposed via *mailserver.mailboxes*, where all default mailboxes are defined.

CHAPTER

ELEVEN

SENDER REWRITING SCHEME

The Sender Rewriting Scheme (SRS) allows mail servers to forward emails without breaking SPF checks. By rewriting the envelope sender to an address within the forwarder's domain, SRS ensures that forwarded messages pass SPF validation, preventing them from being rejected as spoofed or unauthorized.

11.1 How SRS works in practice

1. `alice@foo.example` receives an E-Mail from `bob@bar.example`. Both the envelope sender as well as the `From` header show `bob@bar.example`. This results in strict SPF alignment, because `bar.example` is the domain used in both the `Return-Path` and `FROM` headers.

2. `alice@foo.example` forwards the mail to `charlie@moo.example` and uses SRS to rewrite the envelope sender to originate from the local SRS domain (e.g. *SRS0=HHH=TT=bar.example=alice@foo.example*). The `FROM` header remains unchanged. This ensures that the forwarded mail succeeds SPF checks.

3. The email reaches `charlie@moo.example`. SPF passes because the sender domain in the envelope has been rewritten. The mismatch between envelope sender domain and `FROM` domain does however break strict SPF alignment.

11.2 Enabling SRS

In a simple setup just enabling SRS will use your `mailserver.systemDomain` when rewriting the envelope sender domain.

```
{
  mailserver = {
    srs = {
      enable = true;
      #domain = "srs.example.com";
    };
  };
};
```

While you can reuse an existing email domain for SRS, it is recommended to configure a dedicated SRS domain. This is particularly important under the following conditions:

- Multiple unrelated mail domains are hosted on the mailserver
- The mail domain requires strict SPF alignment in its DMARC policy

11.3 Required DNS changes

Note: In the following example we assume that you want to set up a dedicated SRS domain. If that is not the case you already have SPF and DKIM set up for the system domain. If you have a DMARC record on the system domain, make sure it uses a relaxed SPF alignment policy (`aspf=r`).

First we set up an MX record. This is so that we can receive and route bounces that can result from forwards.

Name (Subdomain)	TTL	Type	Priority	Value
srs.example.com	10800	MX	10	`mail.example.com`

Next up is the SPF record on the SRS domain to allow SPF authentication.

Name (Subdomain)	TTL	Type	Value
srs.example.com	10800	TXT	`v=spf1 mx -all`

Then we deploy the DKIM record with the *p=<value>* taken from `/var/dkim/srs.example.com.mail.txt`, that appears after deploying with SRS enabled.

Name (Subdomain)	TTL	Type	Value
mail._domainkey.srs.example.com	10800	TXT	`v=DKIM1; k=rsa; p=<really-long-key>`

Finally we can tie this together in the DMARC record to require receivers to verify the requested SPF/DKIM alignment.

Note: The SRS domain can only support relaxed SPF alignment due to the envelope sender and `FROM` header mismatch.

Name (Subdomain)	TTL	Type	Value
_dmarc.srs.example.com	10800	TXT	`v=DMARC1; p=reject; aspf=r; adkim=s;`

We can safely configure a `reject` policy on the SRS domain, to enforce the SPF and DKIM alignment as configured above.

CHAPTER

TWELVE

AUTODISCOVERY

RFC6186 defines how email clients can automatically discover a mail server's SMTP and IMAP endpoints. To enable this, the following DNS records must be configured:

Table 1: Resource record set

Name	TTL	Type	Priority	Weight	Port	Value
_submissions._tcp.example.com.	3600	SRV	10	1	465	mail.example.com.
_imaps._tcp.example.com.	3600	SRV	10	1	993	mail.example.com.

12.1 Legacy records

The following DNS records are only supported with *mailserver.enableSubmission* and *mailserver.enableImap*, because they only support connections with explicit TLS. These services are disabled by default because they are deprecated through RFC8314 4.1.

Table 2: Resource record set

Name	TTL	Type	Priority	Weight	Port	Value
_submission._tcp.example.com.	3600	SRV	20	1	587	mail.example.com.
_imap._tcp.example.com.	3600	SRV	20	1	143	mail.example.com.

12.2 Client support

As researched in March 2026

Only a small number of MUAs currently implement this. The most common concern from the bigger and security-conscious vendors is lack of widespread DNSSEC propagation that could be used to authenticate these SRV records.

- Aerc: since 0.20.1
  - `_submissions._tcp` support submitted in https://lists.sr.ht/~rjarry/aerc-devel/patches/68173
- Evolution: Since 3.49.3 for mail accounts
  - https://gitlab.gnome.org/GNOME/evolution/-/wikis/Autoconfig
  - https://gitlab.gnome.org/GNOME/evolution/-/issues/941

12.2.1 Unsupported

- DeltaChat:
  - https://github.com/chatmail/core/issues/1508
- Thunderbird:
  - Desktop: https://bugzilla.mozilla.org/show_bug.cgi?id=342242
  - Android: https://github.com/thunderbird/thunderbird-android/issues/4721

12.3 Vendor-specific autoconfig

The automx2 service can provide autoconfig support for Apple's mobileconfig, Microsoft's Autodiscover and Mozilla's Autoconfig standards. It does however lack support for multiple mail domains and isn't open for contributions due to copyright concerns.

CHAPTER

THIRTEEN

BACKUP GUIDE

First off you should have a backup of your `configuration.nix` file where you have the server config (but that is already in a git repository right?)

Next you need to backup `/var/vmail` or whatever you have specified for the option *mailserver.storage.path*. This is where all the mails reside. Good options are a cron job with `rsync` or `scp`. But really anything works, as it is simply a folder with plenty of files in it. If your backup solution does not preserve the owner of the files don't forget to `chown` them to `virtualMail:virtualMail` if you copy them back (or whatever you specified as *mailserver.storage.owner*, and *mailserver.storage.group*).

To backup spam and ham training data, backup `/var/lib/redis-rspamd`.

Finally you can (optionally) make a backup of `/var/dkim` (or whatever you specified as *mailserver.dkim.keyDirectory*). If you should lose those don't worry, new ones will be created on the fly. But you will need to update the DKIM TXT records to reflect the new key material.

CHAPTER

FOURTEEN

FLAKES

To use NixOS mailserver Nix flakes, the following minimal `flake.nix` can serve as an example to get started:

```
{
  description = "NixOS configuration";

  inputs = {
    nixpkgs.url = "github:NixOS/nixpkgs/nixos-26.05-small";

    nixos-mailserver.url = "gitlab:simple-nixos-mailserver/nixos-mailserver/nixos-26.05";
    nixos-mailserver.inputs.nixpkgs.follows = "nixpkgs";
  };

  outputs =
    {
      nixpkgs,
      nixos-mailserver,
      ...
    }:
    {
      nixosConfigurations = {
        hostname = nixpkgs.lib.nixosSystem {
          system = "x86_64-linux"; # or aarch64-linux
          modules = [
            nixos-mailserver.nixosModules.default
            {
              mailserver = {
                enable = true;

                # Check the setup guide if you have no idea how to continue
                # from here!
              };
            }
          ];
        };
      };
    };
}
```

Lock the inputs and deploy the system closure:

```
nix flake lock
nixos-rebuild --target-host root@mail.example.com --flake .#hostname switch
```

CHAPTER

FIFTEEN

TUNE SPAM FILTERING

SNM comes with the rspamd spam filtering system enabled by default. Although its out-of-the-box performance is good, you can increase its efficiency by tuning its behaviour.

15.1 Auto-learning

Moving spam email to the Junk folder (and false-positives out of it) will trigger an automatic training of the Bayesian filters, improving filtering of future emails.

15.2 Train from existing folders

If you kept previous spam, you can train the filter from it. Note that the rspamd FAQ indicates that *you should always learn both classes with almost equal amount of messages to increase performance of the statistical engine.*

You can run the training in a root shell as follows:

```
# Learn the Junk folder as spam
rspamc learn_spam /var/vmail/$DOMAIN/$USER/.Junk/cur/

# Learn the INBOX as ham
rspamc learn_ham /var/vmail/$DOMAIN/$USER/cur/

# Check that training was successful
rspamc stat | grep learned
```

15.3 Tune symbol weight

The `X-Spamd-Result` header is automatically added to your emails, detailing the scoring decisions. The modules documentation details the meaning of each symbol. You can tune the weight if a symbol if needed.

```
services.rspamd.locals = {
  "groups.conf".text = ''
    symbols "FORGED_RECIPIENTS" {
      weight = 0;
    }
  '';
};
```

15.4 Tune action thresholds

After scoring the message, rspamd decides on an action based on configurable thresholds. By default, rspamd will tell postfix to reject any message with a score higher than 15. If you experience issues in scoring or want to stay on the safe side, you can disable this behaviour by tuning the configuration. For example:

```
services.rspamd.extraConfig = ''
  actions {
    reject = null; # Disable rejects, default is 15
    add_header = 6; # Add header when reaching this score
    greylist = 4; # Apply greylisting when reaching this score
  }
'';
```

15.5 Access the rspamd web UI

Rspamd comes with a web interface that displays statistics and history of past scans. **We do NOT recommend using it to change the configuration** as doing so will override values from the configuration set in the previous sections.

The UI is served on the `/var/run/rspamd/worker-controller.sock` Unix socket. Here are two ways to access it from your browser.

15.5.1 With ssh forwarding

For occasional access, the simplest way is to forward the socket to localhost and open http://localhost:3333 in your browser.

```
ssh -L 3333:/run/rspamd/worker-controller.sock $HOSTNAME
```

15.5.2 With an nginx reverse-proxy

If you have a secured nginx reverse proxy set on the host, you can use it to expose the socket. **Keep in mind the UI is unsecured by default, you need to setup an authentication scheme**, for example with basic auth:

```
services.nginx.virtualHosts.rspamd = {
  forceSSL = true;
  enableACME = true;
  basicAuthFile = "/basic/auth/hashes/file";
  serverName = "rspamd.example.com";
  locations = {
    "/" = {
      proxyPass = "http://unix:/run/rspamd/worker-controller.sock:/";
    };
  };
};
```

CHAPTER

SIXTEEN

RADICALE

Radicale is a lightweight open-source CalDAV/CardDAV server that stores calendars and contacts as plain files on the filesystem, enabling simple self-hosted synchronization with standard clients.

16.1 Limitations

Radicale since the 3.x release (introduced in NixOS 20.09) does not support traditional crypt() password hashes any longer. To establish access for existing *mailserver.accounts*, the hashing method used for `hashedPassword` needs to be compatible with one of the available htpasswd_encryption methods. Such hashes can for example be created using

```
nix-shell -p mkpasswd --command "mkpasswd -m bcrypt"
```

16.2 Code

Configuration contributed by Robert Schütz (@dotlambda).

```
{
  config,
  pkgs,
  lib,
  ...
}:

let
  inherit (lib)
    concatStrings
    flip
    mapAttrsToList
    ;

  mailAccounts = config.mailserver.accounts;
  htpasswd = pkgs.writeText "radicale.users" (
    concatStrings (flip mapAttrsToList mailAccounts (mail: user: "${mail}+:${user.
↪hashedPassword}\n"))
  );

in
{
```

(continues on next page)

(continued from previous page)

```
  services.radicale = {
    enable = true;
    settings = {
      auth = {
        type = "htpasswd";
        htpasswd_filename = "${htpasswd}";
        htpasswd_encryption = "bcrypt";
      };
    };
  };

  services.nginx = {
    enable = true;
    virtualHosts = {
      "cal.example.com" = {
        forceSSL = true;
        enableACME = true;
        locations."/" = {
          proxyPass = "http://localhost:5232/";
          extraConfig = ''
            proxy_set_header  X-Script-Name /;
            proxy_set_header  X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_pass_header Authorization;
          '';
        };
      };
    };
  };

  networking.firewall.allowedTCPPorts = [
    80
    443
  ];
}
```

CHAPTER

SEVENTEEN

ROUNDCUBE

Roundcube is a browser-based open-source webmail client that provides a full-featured email interface with support for IMAP, SMTP, address books, and extensible plugins.

17.1 Code

The NixOS module for Roundcube integrates almost immediately with NixOS mailserver, automatically configuring an Nginx virtual host and ACME-managed TLS for secure webmail access; using other web servers may require additional manual setup.

Once set up you can login with your login account credentials.

```
{
  config,
  pkgs,
  ...
}:

{
  services.roundcube = {
    enable = true;
    hostName = "webmail.example.com"; # the nginx vhost
    package = pkgs.roundcube.withPlugins (
      plugins: with plugins; [
        # external plugins to be included
        # https://search.nixos.org/packages?query=roundcubePlugins
        persistent_login
      ]
    );
    # activate plugins
    plugins = [
      "persistent_login"
      "managesieve" # built-in
    ];
    dicts = with pkgs.aspellDicts; [
      # https://search.nixos.org/packages?query=aspellDicts
      en
    ];
    maxAttachmentSize = config.mailserver.messageSizeLimit / 1024 / 1024;
    extraConfig = ''
```

(continues on next page)

(continued from previous page)

```
      $config['imap_host'] = "ssl://${config.mailserver.fqdn}";
      $config['smtp_host'] = "ssl://${config.mailserver.fqdn}";
      $config['smtp_user'] = "%u";
      $config['smtp_pass'] = "%p";

      $config['managesieve_host'] = "tls://${config.mailserver.fqdn}";
      $config['managesieve_port'] = 4190;
      $config['managesieve_usetls'] = true;
    '';
  };

  services.nginx.virtualHosts.${config.services.roundcube.hostName} = {
    enableACME = true;
    forceSSL = true;
  };

  networking.firewall.allowedTCPPorts = [
    80
    443
  ];
}
```

To use a different reverse proxy, such as Caddy, bind Roundcube's Nginx virtual host to `127.0.0.1` on a custom port and disable SSL and ACME, as the reverse proxy will handle those.

```
{ config, ... }:
{
  services.nginx.virtualHosts.${config.services.roundcube.hostName} = {
    forceSSL = false;
    enableACME = false;
    listen = [
      {
        addr = "127.0.0.1";
        port = 8000;
      }
    ];
  };

  services.caddy.virtualHosts."${config.services.roundcube.hostName}".extraConfig = ''
    reverse_proxy localhost:8000
  '';
}
```

CHAPTER

EIGHTEEN

INDICES AND TABLES

- genindex
- modindex
- search

INDEX

C

M