NixOS Mailserver

Jun 06, 2021

Contents

SNM
Simple Nixos Mailserver

CHAPTER 1

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.

What you need is:

- a server running NixOS with a public IP
- a domain name.

Note: In the following, we consider a server with the public IP `1.2.3.4` and the domain `example.com`.

First, we will set the minimum DNS configuration to be able to deploy an up and running mail server. Once the server is deployed, we could then set all DNS entries required to send and receive mails on this server.

1.1 Setup DNS A record for server

Add a DNS record to the domain `example.com` with the following entries

Name (Subdomain)	TTL	Type	Value
`mail.example.com`	10800	A	`1.2.3.4`

You can check this with

```
$ ping mail.example.com
64 bytes from mail.example.com (1.2.3.4): icmp_seq=1 ttl=46 time=21.3 ms
...
```

Note that it can take a while until a DNS entry is propagated. This DNS entry is required for the Let's Encrypt certificate generation (which is used in the below configuration example).

1.2 Setup the server

The following describes a server setup that is fairly complete. Even though there are more possible options (see the `default.nix` file), these should be the most common ones.

```
{ config, pkgs, ... }:
{
  imports = [
    (builtins.fetchTarball {
      # Pick a commit from the branch you are interested in
      url = "https://gitlab.com/simple-nixos-mailserver/nixos-mailserver/-/archive/A-
↪COMMIT-ID/nixos-mailserver-A-COMMIT-ID.tar.gz";
      # And set its hash
      sha256 = "0000000000000000000000000000000000000000000000000000";
    })
  ];

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

    # A list of all login accounts. To create the password hashes, use
    # nix run nixpkgs.apacheHttpd -c htpasswd -nbB "" "super secret password" | cut -
↪d: -f2
    loginAccounts = {
        "user1@example.com" = {
            hashedPasswordFile = "/a/file/containing/a/hashed/password";
            aliases = ["postmaster@example.com"];
        };
        "user2@example.com" = { ... };
    };

    # Use Let's Encrypt certificates. Note that this needs to set up a stripped
    # down nginx and opens port 80.
    certificateScheme = 3;
  };
}
```

After a `nixos-rebuild switch` your server should be running all mail components.

1.3 Setup all other DNS requirements

1.3.1 Set rDNS (reverse DNS) entry for server

Wherever you have rented your server, you should be able to set reverse DNS entries for the IP's you own. Add an entry resolving `1.2.3.4` to `mail.example.com`

You can check this with

```
$ nix-shell -p bind --command "host 1.2.3.4"
4.3.2.1.in-addr.arpa domain name pointer mail.example.com.
```

Note that it can take a while until a DNS entry is propagated.

1.3.2 Set a MX record

Add a MX record to the domain example.com.

Name (Subdomain)	Type	Priority	Value
example.com	MX	10	mail.example.com

You can check this with

```
$ nix-shell -p bind --command "host -t mx example.com"
example.com mail is handled by 10 mail.example.com.
```

Note that it can take a while until a DNS entry is propagated.

1.3.3 Set a SPF record

Add a SPF record to the domain example.com.

Name (Subdomain)	TTL	Type	Value
example.com	10800	TXT	*v=spf1 a:mail.example.com -all*

You can check this with

```
$ nix-shell -p bind --command "host -t TXT example.com"
example.com descriptive text "v=spf1 a:mail.example.com -all"
```

Note that it can take a while until a DNS entry is propagated.

1.3.4 Set DKIM signature

On your server, the opendkim systemd service generated a file containing your DKIM public key in the file /var/dkim/example.com.mail.txt. The content of this file looks like

```
mail._domainkey IN TXT "v=DKIM1; k=rsa; s=email; p=<really-long-key>" ; ----- DKIM␣
↪mail for domain.tld
```

where really-long-key is your public key.

Based on the content of this file, we can add a DKIM record to the domain example.com.

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

You can check this with

```
$ nix-shell -p bind --command "host -t txt mail._domainkey.example.com"
mail._domainkey.example.com descriptive text "v=DKIM1;p=<really-long-key>"
```

Note that it can take a while until a DNS entry is propagated.

1.3.5 Set a DMARC record

Add a `DMARC` record to the domain `example.com`.

Name (Subdomain)	TTL	Type	Value
_dmarc.example.com	10800	TXT	`v=DMARC1; p=none`

You can check this with

```
$ nix-shell -p bind --command "host -t TXT _dmarc.example.com"
_dmarc.example.com descriptive text "v=DMARC1; p=none"
```

Note that it can take a while until a DNS entry is propagated.

1.4 Test your Setup

Write an email to your aunt (who has been waiting for your reply far too long), and sign up for some of the finest newsletters the Internet has. Maybe you want to sign up for the SNM Announcement List?

Besides that, you can send an email to mail-tester.com and see how you score, and let mxtoolbox.com take a look at your setup, but if you followed the steps closely then everything should be awesome!

CHAPTER 2

Contribute or troubleshoot

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

You can also chat with us on the Libera IRC channel `#nixos-mailserver`.

2.1 Run NixOS tests

You can run the testsuite via

```
$ nix-build tests -A external.nixpkgs_20_03
$ nix-build tests -A internal.nixpkgs_unstable
...
```

2.2 Contributing to the documentation

The documentation is written in RST, build with Sphinx and published by Read the Docs.

For the syntax, see RST/Sphinx Cheatsheet.

The `shell.nix` provides all the tooling required to build the documentation:

```
$ nix-shell
$ cd docs
$ make html
$ firefox ./_build/html/index.html
```

2.3 Nixops

You can test the setup via `nixops`. After installation, do

```
$ nixops create nixops/single-server.nix nixops/vbox.nix -d mail
$ nixops deploy -d mail
$ nixops info -d mail
```

You can then test the server via e.g. `telnet`. To log into it, use

```
$ nixops ssh -d mail mailserver
```

2.4 Imap

To test imap manually use

```
$ openssl s_client -host mail.example.com -port 143 -starttls imap
```

CHAPTER 3

FAQ

3.1 `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.loginAccounts = {
    "user@example.com" = {
        aliases = [ "@example.com" ];
    };
};
```

See also this discussion for details.

CHAPTER 4

Release Notes

4.1 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

4.2 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 5

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 `mailDirectory`. 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 `vmailUserName`, and `vmailGoupName`).

Finally you can (optionally) make a backup of `/var/dkim` (or whatever you specified as `dkimKeyDirectory`). If you should lose those don't worry, new ones will be created on the fly. But you will need to repeat step `B) 5` and correct all the `dkim` keys.

CHAPTER 6

Add Radicale

Configuration by @dotlambda

Starting with Radicale 3 (first introduced in NixOS 20.09) the traditional crypt passwords, as generated by *mkpasswd*, are no longer supported. Instead bcrypt passwords have to be used which can be generated using *htpasswd*.

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

with lib;

let
  mailAccounts = config.mailserver.loginAccounts;
  htpasswd = pkgs.writeText "radicale.users" (concatStrings
    (flip mapAttrsToList mailAccounts (mail: user:
      mail + ":" + user.hashedPassword + "\n"
    ))
  );

in {
  services.radicale = {
    enable = true;
    config = ''
      [auth]
      type = htpasswd
      htpasswd_filename = ${htpasswd}
      htpasswd_encryption = bcrypt
    '';
  };

  services.nginx = {
    enable = true;
    virtualHosts = {
      "cal.example.com" = {
        forceSSL = true;
        enableACME = true;
        locations."/" = {
```

(continues on next page)

(continued from previous page)

```
          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 7

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.

7.1 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.

7.2 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:

```
# Path to the controller socket
export RSOCK="/var/run/rspamd/worker-controller.sock"

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

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

# Check that training was successful
rspamc -h $RSOCK stat | grep learned
```

7.3 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; }
    }'';
};
```

7.4 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
  }
'';
```

7.5 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.

7.5.1 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
```

7.5.2 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 exemple 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 8

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 a plugin to dovecot, `fts_xapian`.

8.1 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;
      # this only applies to plain text attachments, binary attachments are never␣
↪indexed
      indexAttachments = true;
      enforced = "body";
    };
  };
}
```

The `enforced` parameter 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 `autoIndex` to `false`, indices will be created when the IMAP client issues a search query, so latency will be high.

8.2 Resource requirements

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

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

Indexation itself is rather resouces 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 memory limit by eg `mailserver.fullTextSearch.memoryLimit = 2000` (in MiB).

8.3 Mitigating resources requirements

You can:

- disable indexation of attachements `mailserver.fullTextSearch.indexAttachments = false`
- reduce the size of ngrams to be indexed `mailserver.fullTextSearch.minSize` and `maxSize`
- disable automatic indexation for some folders with `mailserver.fullTextSearch.autoIndexExclude`. Folders can be specified by name (`"Trash"`), by special use (`"\\Junk"`) or with a wildcard.

CHAPTER 9

Nix Flakes

If you're using flakes, you can use the following minimal `flake.nix` as an example:

```
{
  description = "NixOS configuration";

  inputs.simple-nixos-mailserver.url = "gitlab:simple-nixos-mailserver/nixos-
↪mailserver/nixos-20.09";

  outputs = { self, nixpkgs, simple-nixos-mailserver }: {
    nixosConfigurations = {
      hostname = nixpkgs.lib.nixosSystem {
        system = "x86_64-linux";
        modules = [
          simple-nixos-mailserver.nixosModule
          {
            mailserver = {
              enable = true;
              # ...
            };
          }
        ];
      };
    };
  };
}
```

CHAPTER 10

Indices and tables

- genindex
- modindex
- search