NixOS Mailserver

Nov 30, 2020

Contents


SNM
Simple Nixos Mailserver

CHAPTER 1

Quick Start

```
{ config, pkgs, ... }:
let release = "nixos-20.09";
in {
  imports = [
    (builtins.fetchTarball {
      url = "https://gitlab.com/simple-nixos-mailserver/nixos-mailserver/-/archive/$
↪{release}/nixos-mailserver-${release}.tar.gz";
      # This hash needs to be updated
      sha256 = "0000000000000000000000000000000000000000000000000000";
    })
  ];

  mailserver = {
    enable = true;
    fqdn = "mail.example.com";
    domains = [ "example.com" "example2.com" ];
    loginAccounts = {
        "user1@example.com" = {
            # mkpasswd -m sha-512 "super secret password" > /hashed/password/file/
↪location
            hashedPasswordFile = "/hashed/password/file/location";

            aliases = [
                "info@example.com"
                "postmaster@example.com"
                "postmaster@example2.com"
            ];
        };
    };
  };
}
```

CHAPTER 2

A Complete 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 `mail-tester.com`.

What you need:

- A server with a public IP (referred to as `server-IP`)
- A Fully Qualified Domain Name (`FQDN`) where your server is reachable, so that other servers can find yours. Common FQDN include `mx.example.com` (where `example.com` is a domain you own) or `mail.example.com`. The domain is referred to as `server-domain` (`example.com` in the above example) and the `FQDN` is referred to by `server-FQDN` (`mx.example.com` above).
- A list of domains you want to your email server to serve. (Note that this does not have to include `server-domain`, but may of course). These will be referred to as `domains`. As an example, `domains = [ example1.com, example2.com ]`.

2.1 A) Setup server

The following describes a server setup that is fairly complete. Even though there are more possible options (see `default.nix`), 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";
    })
  ];
```

(continues on next page)

(continued from previous page)

```
  mailserver = {
    enable = true;
    fqdn = <server-FQDN>;
    domains = [ <domains> ];

    # A list of all login accounts. To create the password hashes, use
    # mkpasswd -m sha-512 "super secret password"
    loginAccounts = {
        "user1@example.com" = {
            hashedPassword = "$6$/z4n8AQl6K
↪$kiOkBTWlZfBd7PvF5GsJ8PmPgdZsFGN1jPGZufxxr60PoR0oUsrvzm2oQiflyz5ir9fFJ.d/zKm/
↪NgLXNUsNX/";

            aliases = [
                "postmaster@example.com"
                "postmaster@example2.com"
            ];

            # Make this user the catchAll address for domains example.com and
            # example2.com
            catchAll = [
                "example.com"
                "example2.com"
            ];
        };

        "user2@example.com" = { ... };
    };

    # Extra virtual aliases. These are email addresses that are forwarded to
    # loginAccounts addresses.
    extraVirtualAliases = {
        # address = forward address;
        "abuse@example.com" = "user1@example.com";
    };

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

    # Enable IMAP and POP3
    enableImap = true;
    enablePop3 = true;
    enableImapSsl = true;
    enablePop3Ssl = true;

    # Enable the ManageSieve protocol
    enableManageSieve = true;

    # whether to scan inbound emails for viruses (note that this requires at least
    # 1 Gb RAM for the server. Without virus scanning 256 MB RAM should be plenty)
    virusScanning = false;
  };
}
```

After a `nixos-rebuild switch --upgrade` your server should be good to go. If you want to use `nixops` to

deploy the server, look in the subfolder `nixops` for some inspiration.

2.2 B) Setup everything else

2.2.1 Step 1: Set DNS entry for server

Add a DNS record to the domain `server-domain` with the following entries

Name (Subdomain)	TTL	Type	Priority	Value
`server-FQDN`	10800	A		`server-IP`

This resolves DNS queries for `server-FQDN` to `server-IP`. You can test if your setting is correct by

```
ping <server-FQDN>
```

Expected output:

```
64 bytes from <server-FQDN> (<server-IP>): icmp_seq=1 ttl=46 time=21.3 ms
...
```

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

2.2.2 Step 2: 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 `server-IP` to `server-FQDN`

You can test if your setting is correct by

```
host <server-IP>
```

Expected output:

```
<server-IP-octets-reversed>.in-addr.arpa domain name pointer <server-FQDN>.
```

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

2.2.3 Step 3: Set `MX` Records

For every `domain` in `domains` do: * Add a `MX` record to the domain `domain`

```
| Name (Subdomain) | TTL   | Type | Priority | Value              |
| ---------------- | ----- | ---- | -------- | ------------------ |
| `domain`         |       | MX   | 10       | `server-FQDN`      |
```

You can test this via

```
dig -t MX <domain>
```

Expected output:

```
...
;; ANSWER SECTION:
<domain>    10800   IN  MX  10 <server-FQDN>
...
```

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

2.2.4 Step 4: Set SPF Records

For every `domain` in `domains` do: * Add a `SPF` record to the domain `domain`

```
| Name (Subdomain) | TTL   | Type | Priority | Value                          |
| ---------------- | ----- | ---- | -------- | ------------------             |
| `domain`         | 10800 | TXT  |          | `v=spf1 ip4:<server-IP> -all` |
```

You can check this with `dig -t TXT <domain>` similar to the last section. Note that `SPF` records are set as `TXT` records since RFC1035.

Note that it can take a while until a DNS entry is propagated. If you want to use multiple servers for your email handling, don't forget to add all server IP's to this list.

2.2.5 Step 5: Set DKIM signature

In this section we assume that your `dkimSelector` is set to `mail`. If you have a different selector, replace all `mail`'s below accordingly.

For every `domain` in `domains` do: * Go to your server and navigate to the dkim key directory (by default `/var/dkim`). There you will find a public key for any domain in the `domain.txt` file. It will look like `mail._domainkey IN TXT "v=DKIM1; r=postmaster; g=*; k=rsa; p=<really-long-key>" ; ----- DKIM mail for domain.tld` * Add a `DKIM` record to the domain `domain`

```
| Name (Subdomain)          | TTL   | Type | Priority | Value                                ␣
↪|
| ----------------          | ----- | ---- | -------- | ------------------                   ␣
↪|
| mail._domainkey.`domain` | 10800 | TXT  |          | `v=DKIM1; p=<really-long-key>`␣
↪|
```

You can check this with `dig -t TXT mail._domainkey.<domain>` similar to the last section.

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

2.2.6 Step 6: Set DMARC record

For every `domain` in `domains` do:

- Add a `DMARC` record to the domain `domain`

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

You can check this with `dig -t TXT _dmarc.<domain>` similar to the last section.

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

2.3 C) 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 3

How to Develop SNM

3.1 Run NixOS tests

You can run the testsuite via

```
$ nix-build tests -A extern.nixpkgs_20_03
$ nix-build tests -A intern.nixpkgs_unstable
...
```

3.2 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
```

3.3 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
```

3.4 Imap

To test imap manually use

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

CHAPTER 4

A Complete Backup Guide

This is really easy. 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 5

How to Add Radicale to SNM

Configuration by @dotlambda

```
{ 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 = crypt
    '';
  };

  services.nginx = {
    enable = true;
    virtualHosts = {
      "cal.example.com" = {
        forceSSL = true;
        enableACME = true;
        locations."/" = {
          proxyPass = "http://localhost:5232/";
          extraConfig = ''
```

(continues on next page)

(continued from previous page)

```
            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 6

How to 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.

6.1 A) 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.

6.2 B) 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
```

6.3 C) 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; }
    }'';
};
```

6.4 D) 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
  }
'';
```

6.5 E) 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.

6.5.1 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
```

6.5.2 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 7

Release Notes

7.1 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 8

Indices and tables

- genindex
- modindex
- search