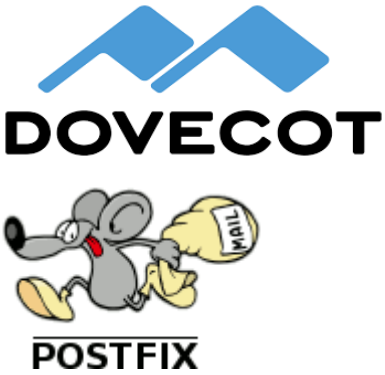
Setting up redundant mail servers and integrate with Nextcloud mail app

Nextcloud has a fantastic mail-app that can be used as a web interface for most e-mail providers. And of course, if you are running Nextcloud, you might want to host your own imap and smtp servers as well. In this blog post I will describe how you can set up and configure dovecot and postfix as redundant imap and smtp servers and how you can integrate them with Nextclouds app-password database so you get all user management for free and your users can have self service password management.



Architectural overview

To get this working you will need a minimum of two servers, where you can run docker containers. You will also need the ability to configure two (sub) domains for use with your e-mail services. In this example we will set up imap.example.com and smtp.example.com on four servers, that means you should replace any instance of 'example.com' below with your own domain:

1. imap1.example.com
2. imap2.example.com
3. smtp1.example.com
4. smtp2.example.com

If you are so inclined, you cound of course consolidate this to only two servers, e.g.

- mail1.example.com
- mail2.example.com

We will use docker-compose to manage the containers.

You will also need some kind of loadbalancing, that can load balance on tcp ports to get high availability for this set up. One option is to use round robin dns or even have one of the servers be a hot standby that you can switch to by updating the dns record (make sure you have a short TTL set for your dns records in that case). I will not go in to a lot of details on how you can set up load balancing, but you should be able use your favourite solution for this.

The same thing is assumed for certificates, you need to have a certificate called **/opt/dovecot/certs/imap.example.com/fullchain.pem** and a key called **/opt/dovecot/certs/imap.example.com/privkey.pem** on the imap servers, and similarly a certificate called **/opt/postfix/certs/smtp.example.com/fullchain.pem** and a key called **/opt/postfix/certs/smtp.example.com/privkey.pem** on the smtp servers. I will not cover how to get them and how to distribute them to your server here, but I encourage you to look at certbot if you don't know where to go from there.

Companion docker containers

SUNET builds two container images from these repositories, that can be used with this guide:

- https://github.com/SUNET/dovecot-lda-docker
- https://github.com/SUNET/postfix-docker

These images can be pulled from:

- docker.sunet.se/mail/dovecot
- docker.sunet.se/mail/postfix

Common configuration

This guide will assume you are running Debian 12 on your machines, but you should be able to adjust the commands here to any modern linux distro. At SUNET we use a rather fancy puppet setup to manage our servers, but I will describe installation commands using apt command line syntax below. In general you will need to be root for these commands, so either prefix with sudo or make sure you are running as the root user. The first thing we need is docker and docker-compose:

```
# apt update && apt install docker.io docker-compose
```

This needs to be done **on all servers**.

Dovecot

This needs to be done **on imap1.example.com and imap2.example.com.**

Next we will set up dovecot. Let's first create a directory structure on our host:

```
# mkdir -p /opt/dovecot/{certs,config,mail,ssmtp}
```

Next create the file **/opt/dovecot/docker-compose.yml** with the following content:

/opt/dovecot/docker-compose.yml

```
version: "3.7"
services:
  dovecot:
    image: docker.sunet.se/mail/dovecot:SUNET-1
    volumes:
      - /opt/dovecot/ssmtp/ssmtp.conf:/etc/ssmtp/ssmtp.conf
      - /opt/dovecot/config:/etc/dovecot/
      - /opt/dovecot/mail:/var/mail/
      - /opt/dovecot/certs/:/certs
    command:
      - /usr/sbin/dovecot
      - -F
    ports:
      - "24:24"
      - "143:143"
      - 993:993
      - 4190:4190
      - 12345:12345
      - 12346:12346
    restart: always
```

Next you will need some dovecot configuration:

/opt/dovecot/config/dovecot.conf

```
mail_home=/srv/mail/%Lu
mail_location=maildir:/var/mail/vhosts/%d/%n/
mail_privileged_group = mail
log_path=/dev/stdout
first_valid_uid=8
postmaster_address = postmaster@example.com
sendmail_path = /usr/sbin/ssmtp

namespace inbox {
  inbox = yes
  separator = /
  mailbox Drafts {
    special_use = \Drafts
```

```
    auto = subscribe
  }
  mailbox Junk {
    special_use = \Junk
    auto = subscribe
  }
  mailbox Trash {
    special_use = \Trash
    auto = subscribe
  }

  mailbox Sent {
    special_use = \Sent
    auto = subscribe
  }
  mailbox "Sent Messages" {
    special_use = \Sent
    auto = subscribe
  }
}

protocols = imap lmtp sieve

service lmtp {
   inet_listener lmtp {
      address = 0.0.0.0 ::
      port = 24
   }
}

mail_plugins = mail_plugins notify replication
protocol lmtp {
  mail_plugins = mail_plugins sieve
}
protocol sieve {
  mail_debug = yes
  managesieve_max_line_length = 65536
}

service managesieve-login {
  inet_listener sieve {
    port = 4190
    ssl = yes
  }
  service_count = 1
}
service managesieve {
  process_limit = 256
}

plugin {
   sieve = ~/.dovecot.sieve
   sieve_global_path = /var/lib/dovecot/sieve/default.sieve
   sieve_dir = ~/sieve
   sieve_global_dir = /var/lib/dovecot/sieve/
   sieve_extensions = +vacation-seconds
   sieve_global_extensions = +vnd.dovecot.pipe
   sieve_pipe_bin_dir = /etc/dovecot/sieve
   sieve_plugins = sieve_imapsieve sieve_extprograms
   sieve_vacation_default_period = 7d
   sieve_vacation_max_period = 30d
   sieve_vacation_min_period = 1d
}


auth_mechanisms = plain login
auth_username_format = %n
passdb {
  args = password=<a secret you need for nextcloud goes here> allow_nets=<a comma separated list of your
nextcloud servers goes here>
  driver = static
```

```
}

passdb {
  driver = lua
  args = file=/etc/dovecot/nextcloud-auth.lua
}

userdb {
  driver = sql
  args = /etc/dovecot/dovecot-sql.conf
}

service auth {
 inet_listener {
   port = 12346
 }
}

auth_debug = yes
auth_verbose = yes

ssl=yes
ssl_cert=</certs/imap.example.com/fullchain.pem
ssl_key=</certs/imap.example.com/privkey.pem

doveadm_password = <a shared secret for replicating between the two servers goes here>
service replicator {
  process_min_avail = 1
}
service aggregator {
  fifo_listener replication-notify-fifo {
    user = mail
  }
  unix_listener replication-notify {
    user = mail
  }
}
service doveadm {
  inet_listener {
    port = 12345
  }
}
plugin {
  mail_replica = tcp:<host name of the partner server goes here>:12345
}
```

/opt/dovecot/config/dovecot-sql.conf

```
driver = mysql
connect = host=<database hostname goes here> dbname=<nextcloud database name goes here> user=<nextcloud db
user> password=<nextcloud db password>
user_query = SELECT '%n' as username, 'mail' as uid, 'mail' as gid, '/var/mail/vhosts/example.com/%n' as
home, 'maildir:/var/mail/vhosts/example.com/%n/' as mail;
iterate_query = SELECT UNIQUE(REPLACE(value, '@example.com', '')) AS username, 'example.com' as domain FROM
oc_accounts_data WHERE name = 'email' AND value LIKE '%%example.com';
```

The lua script below will do the actual validation of app passwords, but in stead of implementing sha512 in lua, we will just call out to php and do the exact same thing Nextcloud does when validating app passwords.

This is the one place where you can not have multiple database servers configured, so if you want High Availabilty for this part, you will need to look in to proxysql. You could for instance run proxysql locally in docker on the imap server, and then connect to that container from the lua script to gain high availability.

/opt/dovecot/config/nextcloud-auth.lua

```
function auth_passdb_lookup(req)
  -- Get the hash out using php
  local salt = "<the salt from config.php>"
  local command = "php -r " .. "'print(hash(" .. '"sha512","' .. req.password .. salt .. '"' .. "));'"
  local handle = assert(io.popen(command))
  local hash = handle:read("*a")
  handle:close()

  -- Get the stored app passwords from Nextcloud
  local db        = '<nextcloud database name goes here>'
  local user      = '<nextcloud db user>'
  local password  = '<nextcloud db password>'
  local db_server = '<database hostname goes here>'
  local mysql     = require "luasql.mysql"
  -- Adjust the query below so that you can find your users if the don't have usernames like user@example.com
  local query     = "SELECT token FROM oc_authtoken where uid = '" .. req.user .. "@example.com'"
  local env       = assert(mysql.mysql())
  local conn      = assert(env:connect(db, user, password, db_server))
  local cur       = assert(conn:execute(query))
  local row       = cur:fetch({}, "a")
  while row do
    local token = row.token
    if token == hash then
      return dovecot.auth.PASSDB_RESULT_OK, "password=" .. req.password
    end
    row = cur:fetch(row, "a")
  end
  return dovecot.auth.PASSDB_RESULT_USER_UNKNOWN, "no such user"
end
```

/opt/dovecot/config/ssmtp.conf

```
root=postmaster
mailhub=smtp.example.com:25
rewriteDomain=example.com
hostname=<host name of current imap server>
```

⚠ The configuration in <brackets> in the configuration needs to be replaced with actual values:

- **<a secret you need for nextcloud goes here> This is your master password you need for confguring in Nextcloud, you can generate a strong and complicated password for this**
- **<a comma separated list of your nextcloud servers goes here> Any ip address mentioned here can access any email using the master password above**
- **<a shared secret for replicating between the two servers goes here> This is the authentication token used for replication between the two imap servers, you can generate a strong and complicated password for this**
- **<database hostname goes here> The host name of the nextcloud database server, the `host=` stanza can be repeted in the dovecot-sql.conf file (but not in the lua-script) if you have multiple database hosts in a cluster**
- **<nextcloud database name goes here> The name of the nextcloud database, usually `nextcloud`**
- **<nextcloud db user> The name of your nextcloud database user**
- **<nextcloud db password> The password of your nextcloud database user**
- **<host name of the partner server goes here> This is the other servers host name, i.e. imap2.example.com or imap1.example.com**
- **<the salt from config.php> This is a secret generated by nextcloud and stored in config.php that you need to share with your imap servers**
- **<host name of current imap server> This is this servers host name, i.e. imap1.example.com or imap2.example.com**

Postfix

This needs to be done **on smtp1.example.com and smtp2.example.com.**

First create the required directory structure:

```
# mkdir -p /opt/postfix/{certs,config}
```

Now create /opt/postfix/docker-compose.yml

/opt/postfix/docker-compose.yml

```
version: "3.7"
services:
  postfix:
    image: docker.sunet.se/mail/postfix:SUNET-1
    volumes:
      - /opt/postfix/config:/config
      - /opt/postfix/certs/:/certs
    command:
      - /start.sh
    ports:
      - "25:25"
      - 587:587
    restart: always
```

Then create the /opt/postfix/config/main.cf file:

```
maillog_file = /dev/stdout
smtpd_banner = $myhostname ESMTP $mail_name (Debian/GNU)
biff = no
append_dot_mydomain = no
readme_directory = no
compatibility_level = 3.6
smtpd_tls_cert_file=/certs/smtp.example.com/fullchain.pem
smtpd_tls_key_file=/certs/smtp.example.com/privkey.pem
smtpd_tls_security_level=may
smtp_tls_CApath=/etc/ssl/certs
smtp_tls_security_level=may
smtp_tls_session_cache_database = btree:${data_directory}/smtp_scache
smtpd_relay_restrictions = permit_mynetworks permit_sasl_authenticated defer_unauth_destination
smtpd_client_restrictions = permit_mynetworks
myhostname = <current smtp hostname goes here>
alias_maps = hash:/etc/aliases
alias_database = hash:/etc/aliases
virtual_mailbox_domains = mysql:/config/mysql-virtual-mailbox-domains.cf
virtual_mailbox_maps = mysql:/config/mysql-virtual-mailbox-maps.cf
mydestination = $myhostname, localhost.localdomain, localhost
mynetworks = 127.0.0.0/8 [::ffff:127.0.0.0]/104 [::1]/128 <ip address of imap servers goes here>
mailbox_size_limit = 0
recipient_delimiter = +
inet_interfaces = all
inet_protocols = all
relayhost = <a comma seprated list of relay servers>
smtpd_sasl_type = dovecot
smtpd_sasl_path = inet:imap.example.com:12346
smtpd_sasl_auth_enable = yes
virtual_transport = lmtp:imap.example.com:24
```

Add the /opt/postfix/config/master.cf file

/opt/postfix/config/master.cf

```
smtp      inet  n       -       n       -       -       smtpd
submission inet n       -       n       -       -       smtpd
  -o syslog_name=postfix/submission
  -o smtpd_tls_security_level=encrypt
  -o smtpd_sasl_auth_enable=yes
  -o smtpd_tls_auth_only=yes
  -o smtpd_reject_unlisted_recipient=no
pickup    unix  n       -       n       60      1       pickup
cleanup   unix  n       -       n       -       0       cleanup
qmgr      unix  n       -       n       300     1       qmgr
tlsmgr    unix  -       -       n       1000?   1       tlsmgr
rewrite   unix  -       -       n       -       -       trivial-rewrite
bounce    unix  -       -       n       -       0       bounce
defer     unix  -       -       n       -       0       bounce
trace     unix  -       -       n       -       0       bounce
verify    unix  -       -       n       -       1       verify
flush     unix  n       -       n       1000?   0       flush
proxymap  unix  -       -       n       -       -       proxymap
proxywrite unix -       -       n       -       1       proxymap
smtp      unix  -       -       n       -       -       smtp
relay     unix  -       -       n       -       -       smtp
        -o syslog_name=postfix/$service_name
showq     unix  n       -       n       -       -       showq
error     unix  -       -       n       -       -       error
retry     unix  -       -       n       -       -       error
discard   unix  -       -       n       -       -       discard
local     unix  -       n       n       -       -       local
virtual   unix  -       n       n       -       -       virtual
lmtp      unix  -       -       n       -       -       lmtp
anvil     unix  -       -       n       -       1       anvil
scache    unix  -       -       n       -       1       scache
postlog   unix-dgram n  -       n       -       1       postlogd
maildrop  unix  -       n       n       -       -       pipe
  flags=DRXhu user=vmail argv=/usr/bin/maildrop -d ${recipient}
uucp      unix  -       n       n       -       -       pipe
  flags=Fqhu user=uucp argv=uux -r -n -z -a$sender - $nexthop!rmail ($recipient)
ifmail    unix  -       n       n       -       -       pipe
  flags=F user=ftn argv=/usr/lib/ifmail/ifmail -r $nexthop ($recipient)
bsmtp     unix  -       n       n       -       -       pipe
  flags=Fq. user=bsmtp argv=/usr/lib/bsmtp/bsmtp -t$nexthop -f$sender $recipient
scalemail-backend unix -       n       n       -       2       pipe
  flags=R user=scalemail argv=/usr/lib/scalemail/bin/scalemail-store ${nexthop} ${user} ${extension}
mailman   unix  -       n       n       -       -       pipe
  flags=FRX user=list argv=/usr/lib/mailman/bin/postfix-to-mailman.py ${nexthop} ${user}
```

Add the/opt/postfix/config/mysql-virtual-mailbox-domains.cf file

/opt/postfix/config/mysql-virtual-mailbox-domains.cf

```
user = <nextcloud db user>
password = <nextcloud db password>
hosts = <nextcloud db hosts>
dbname = <nextcloud db name>
query = SELECT 1 WHERE 'example.com' = '%s'
```

/opt/postfix/config/mysql-virtual-mailbox-maps.cf

```
user = <nextcloud db user>
password = <nextcloud db password>
hosts = <nextcloud db hosts>
dbname = <nextcloud db name>
query = SELECT UNIQUE(1) FROM oc_accounts_data WHERE value='%s' and name = 'email'
```

The configuration in <brackets> in the configuration needs to be replaced with actual values:

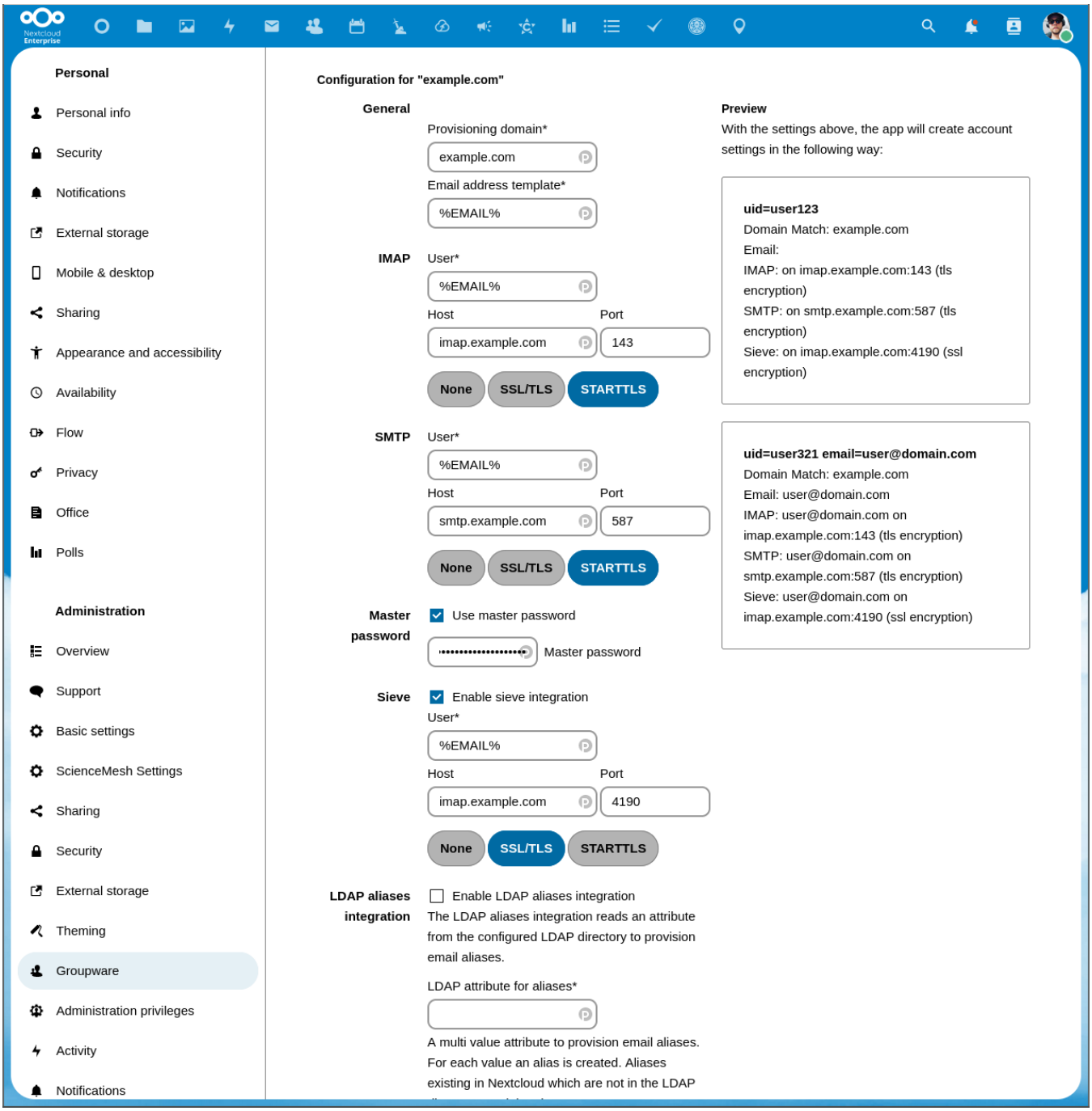
- **<ip address of imap servers goes here> add the ip addresses space separated here to allow them to send vacation responders and such**
- **<a comma seprated list of relay servers> if you are using any relay servers to send your emails add them here**
- **<nextcloud db user> User name of the Nextcloud db user**
- **<nextcloud db password> Password of the Nextcloud db user**
- **<nextcloud db hosts> A space separated list of database servers for Nextcloud (if you have more than one, otherwise just the one then)**
- **<nextcloud db name> The name of the Nextcloud database**

Using Nextcloud mail with SSO-accounts

It is currently not possible to use the Nextcloud mail app with accounts that have logged in using Single Sign On. That is because Nextcloud expects the Nextcloud password to be used for authenticating against the email server (unless you have a gmail or outlook account with ouath set up). However, that is about to change. I have written a PR for the Nextcloud mail app that allows you set a "master password" for the provisioned accounts, when that PR is merged it will be possible to use the Nextcloud mail app with the configuration in this guide.

Configuring Nextcloud

Make sure you enable the mail app in Nextcloud , as an administrator you should then be able to create a "provisioning" in Admin settings Groupware. Note that in this screen shot we are running the patched version that can use master passwords.


Personal
Personal info
Security
Notifications
External storage
Mobile & desktop
Sharing
Appearance and accessibility
Availability
Flow
Privacy
Office
Polls
Administration
Overview
Support
Basic settings
ScienceMesh Settings
Sharing
Security
External storage
Theming
Groupware
Administration privileges
Activity
Notifications
Configuration for "example.com"
General
Provisioning domain*
example.com
Email address template*
%EMAIL%
IMAP
User*
%EMAIL%
Host
Port
imap.example.com
143
None
SSL/TLS
STARTTLS
SMTP
User*
%EMAIL%
Host
Port
smtp.example.com
587
None
SSL/TLS
STARTTLS
Master password
Use master password
Master password
Sieve
Enable sieve integration
User*
%EMAIL%
Host
Port
imap.example.com
4190
None
SSL/TLS
STARTTLS
LDAP aliases integration
Enable LDAP aliases integration
The LDAP aliases integration reads an attribute from the configured LDAP directory to provision email aliases.
LDAP attribute for aliases*
A multi value attribute to provision email aliases. For each value an alias is created. Aliases existing in Nextcloud which are not in the LDAP
Preview
With the settings above, the app will create account settings in the following way:
uid=user123
Domain Match: example.com
Email:
IMAP: on imap.example.com:143 (tls encryption)
SMTP: on smtp.example.com:587 (tls encryption)
Sieve: on imap.example.com:4190 (ssl encryption)
uid=user321 email=user@domain.com
Domain Match: example.com
Email: user@domain.com
IMAP: user@domain.com on imap.example.com:143 (tls encryption)
SMTP: user@domain.com on smtp.example.com:587 (tls encryption)
Sieve: user@domain.com on imap.example.com:4190 (ssl encryption)