Почта

Рассматриваем настройку контейнерезированного почтового сервиса на основе mailcow, а также факторы, влияющие на доставку почты.

Оглавление

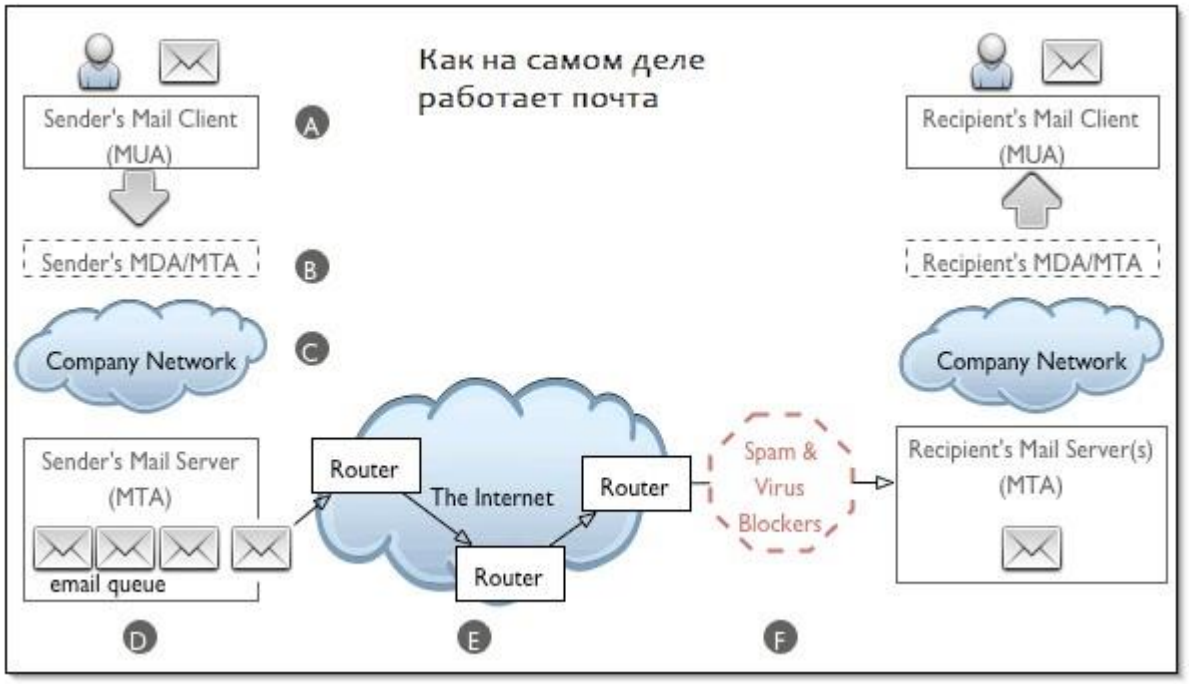
Принцип работы почты

Сейчас легко получить готовое решение с почтой в облаке, но одна из основных причин развертывания почты внутри своей инфраструктуры — контроль. Сохраняя письма внутри своей инфраструктуры, вы можете отправлять письма пользователям внутри организации, не передавая их через интернет.

Кажется, что все просто

С точки зрения пользователя электронная почта кажется достаточно простой. Вы набираете адрес электронной почты человека, которому хотите отправить письмо, пишете сообщение и нажимаете кнопку «Отправить».

В реальности же процесс гораздо более сложный и включает большое количество компонентов, которые оказывают влияние на доставку почты



- **A. Отправитель создает электронное письмо.** Отправитель создает электронное письмо в своём почтовом клиенте (Mail User Agent, MUA) и нажимает кнопку «Отправить». MUA — приложение, которое отправитель использует для составления и чтения электронной почты, например Thunderbird, Outlook, The Bat и так далее. Кроме этого, MUA может быть Web-Based и по факту быть веб-страницей в браузере.
- **B. MTA отправителя маршрутизирует почту.** Mail User Agent (MUA) отправителя передает электронную почту агенту доставки почты (Mail Delivery Agent, MDA). Зачастую агент передачи почты (Mail Transfer Agent, MTA) отправителя также выполняет обязанности MDA, но это могут быть разные серверы. Наиболее популярные MTA — Semdmail, Postfix, Exim. MTA-сервер в основном занимается передачей почты другим почтовым серверам, MDA же направляет письма конечным пользователям. На приведенной диаграмме MDA получает письмо от пользователя и через облако корпоративной сети передачи данных передаёт это письмо MTA.
- **C. Облако сети передачи данных.** Электронная почта может быть внутри корпоративной сети передачи данных или хостинг-провайдера, или же просто размещена в интернете. Сетевое облако может охватывать множество почтовых серверов, DNS-серверов, маршрутизаторов и других устройств и сервисов, которые мы представляем в виде облака. Все эти компоненты склонны быть медленными при обработке достаточно большого количества запросов, временно недоступны для обработки запросов, неверно сконфигурированы, из-за чего другие MTA в сети не могут доставить почту в соответствии с адресом получателя. Эти устройства могут быть защищены файерволом, спам-фильтрами и программным обеспечением для обнаружения вредоносных программ, которые могут запретить доставку или даже удалить электронное письмо. Если это происходит, то, как правило, отправителю не предоставляется информация, где или когда произошел сбой доставки.

  Поставщики услуг электронной почты и другие компании, обрабатывающие большой объем электронной почты, часто имеют несколько почтовых серверов и маршрутизируют всю электронную почту через центральный сервер шлюза (почтовый хаб), который перераспределяет почту на любой доступный MTA. Электронная почта на этих вторичных MTA обычно должна ожидать в очереди, пока основной MTA не станет доступным для обработки

письма, и только после этого вторичный почтовый сервер передаст сообщения первичному MTA.

- **D. Почтовая очередь.** В нашем случае, почта адресована в другой домен, так что MTA должен отправить её за пределы своего почтового домена. В этот момент письмо помещается в очередь на отправку, так как перед этим конкретным письмом могут быть еще и другие неотправленные письма. Поэтому MTA обрабатывает их по очереди по мере поступления.
- **E. Передача почты между MTA.** В случае с передачей почты между MTA, почтовый сервер отправителя занимается всеми аспектами доставки почты, до тех пор, пока письмо не было либо получено, либо отклонено MTA получателя. Как только письмо покинуло электронную очередь и попало в Интернет, оно может проходить через цепочку MTA до тех пор, пока не достигнет ответственного за домен получателя. Каждый из MTA в этой цепочке должен получить письмо и найти, кому передавать его. Для этого MTA использует службу доменных имен DNS и конкретный маршрут передачи почты зависит от доступности серверов, ответов от DNS и т. д. DNS нужен, так как MTA отправителя знает только домен получателя, но не знает, какому именно серверу (IP-адресу MTA получателя) надо передавать письмо.

  Чтобы найти IP-адрес сервера получателя, MTA отправителя обращается к DNS, при этом он запрашивает конкретный тип записи — MX (Mail Exchange), которая указывает на почтовый сервер, ответственный за этот домен. DNS-сервер отвечает MX-записью — списком MX-серверов для этого домена, а также приоритетом для каждого из серверов.

  MTA отправителя устанавливает соединение с MX-сервером получателя. MX-сервер — это MTA-сервер, который принимает почту от других доменов (но не всегда занимается отправкой почты для этого домена).

  Далее MTA отправителя спрашивает, может ли MX принять электронное письмо от определённого пользователя, указывает его домен (username@somedomain.tld) и пересылает само письмо.

- **F. Проверка на SPAM, вирусы и т. д.** Электронная почта может быть пропущена через фильтры нежелательной почты (спам-фильтры), а также проверена на наличие вирусов и другого вредоносного содержания. Это обычно происходит до того, как почта передается внутрь домена, и письма, не прошедшие проверку, помещаются в карантин, о чем уведомляется отправитель и, возможно, получатель. Спам-почта обычно удаляется без каких-либо предупреждений. Спам достаточно трудно обнаружить, так как он может принимать очень много различных форм и видов, поэтому, спам-фильтры проверяют широкий набор критериев и склонны ошибаться.
- **G. Доставка письма.** Если письмо смогло дойти до MTA-сервера домена получателя, не было заблокировано антивирусной проверкой или удалено спам-фильтром, оно передается MDA-серверу для доставки в почтовый ящик получателя, где находится, пока получатель не решит, что делать с этим сообщением.

Прием и передача почты

Различают протоколы взаимодействия пользователя с почтой, а также почтовых серверов друг с другом.

MTA-протоколы

MTA должны иметь возможность передачи писем друг другу, для этого их взаимодействие должно быть стандартизировано. MTA могут использовать следующие протоколы взаимодействия:

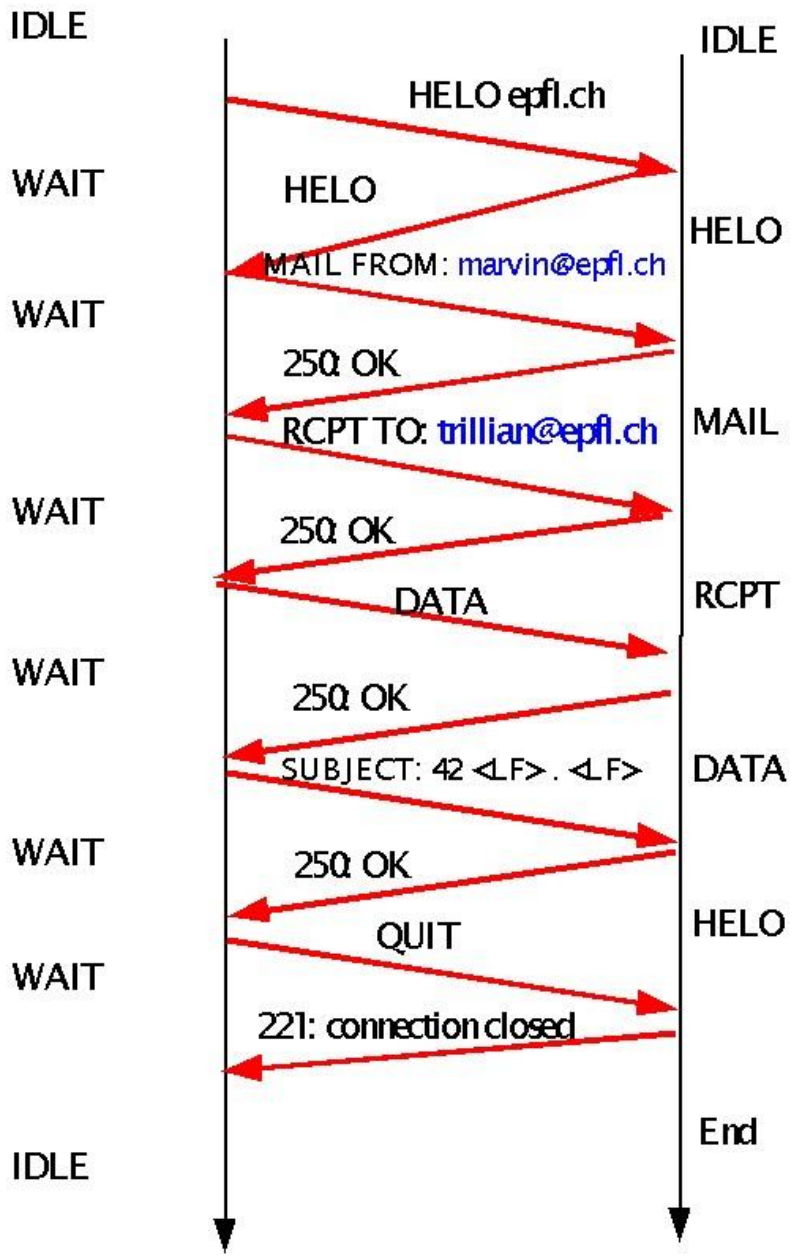
SImple Mail Transfer Protocol (SMTP)

Simple Mail Transfer Protocol (SMTP) был разработан как основной протокол передачи почты через интернет между MTA-серверами. Любой хост, подключенный к интернету, может использовать протокол SMTP для отправки сообщений любому другому хосту.

SMTP использует набор простых команд для установки соединения и передачи информации между хостами. Команды состоят из одного слова (поэтому название протокола начинается со слова Simple), после команды следует дополнительная информация, например:

```
MAIL FROM: <user@somedomain.tld>
```

Это команда удаленному MTA говорит, кто отправитель письма. Длина каждой команды — 4 символа, они передаются открытым ASCII-текстом, и удаленный сервер должен ответить трехзначным кодом, который необходим, чтобы понять, была команда выполнена успешно или нет.



Вы можете сами попробовать отправить письмо, используя утилиту telnet, и посмотреть на взаимодействие. SMTP использует TCP-порт 25:

```
$ telnet mx.yandex.ru 25 Trying 213.180.204.89...

Connected to mx.yandex.ru.

Escape character is '^]'.

220 mxfront8g.mail.yandex.net (Want to use Yandex.Mail for your domain? Visit http://pdd.yandex.ru)

HELO nodrop.in

250 mxfront8g.mail.yandex.net
```

```
MAIL FROM:dradchuk@nodrop.in

250 2.1.0 <dradchuk@nodrop.in> ok

RCPT TO: lliopt@yandex.ru

250 2.1.5 <lliopt@yandex.ru> recipient ok

DATA

354 Enter mail, end with "." on a line by itself

test data.

.

250 2.0.0 Ok: queued on mxfront8g.mail.yandex.net as 1559488408-dpLCIZRTYp-Co1egp62

QUIT

221 Bye
```

Так как мы отправляли письмо с локального компьютера, Яндекс передал его пользователю, но пометил как спам, так как получено оно было от неизвестного MTA 44-179-145-85.ftth.glasoperator.nl, и это не совпадает с доменом, от имени которого мы отправляем письмо (nodrop.in). Но об этом позже.

```
Received: from mxfront8g.mail.yandex.net ([127.0.0.1]) by
      mxfront8g.mail.yandex.net with LMTP id T4z5akXT for
      <lliopt@yandex.ru>; Sun, 2 Jun 2019 18:13:27 +0300
Received:               from               44-179-145-85.ftth.glasoperator.nl
(44-179-145-85.ftth.glasoperator.nl [85.145.179.44])
      by    mxfront8g.mail.yandex.net    (nwsmtp/Yandex)    with    SMTP    id
dpLCIZRTYp-Co1egp62;
      Sun, 02 Jun 2019 18:13:15 +0300
Return-Path: dradchuk@nodrop.in
X-Yandex-Front: mxfront8g.mail.yandex.net
X-Yandex-TimeMark: 1559488395.994
Message-Id: <20190602181327.Co1egp62@mxfront8g.mail.yandex.net>
Date: Sun, 02 Jun 2019 18:13:27 +0300
From: MAILER-DAEMON
To: undisclosed-recipients:;
Authentication-Results:          mxfront8g.mail.yandex.net;          spf=fail
(mxfront8g.mail.yandex.net: domain of nodrop.in does not designate 85.145.179.44
as permitted sender, rule=[-all]) smtp.mail=dradchuk@nodrop.in
X-Yandex-Spam: 4
X-Yandex-Fwd: NDMzMDQ1NDg4NDUzODE2MDk5Myw0ODk5OTg1NDM3NzY1MjYxMzU1
X-Yandex-Forward: 94750db768f64c7630d013a4e9c23fbe
```

Из этого примера видно, что что SMTP-сервер принимает простые текстовые команды и отвечает кодом ответа на каждую из них. RFC 5321 описывает работу протокола SMTP.

Базовые SMTP-команды

Как только TCP-соединение установлено, SMTP-сервер отвечает SMTP-клиенту баннером приветствия Welcome Banner (в приведенном примере 220 mxfront8g.mail.yandex.net), после чего SMTP-клиент может приступать к попытке отправить письмо, используя набор команд:

```
HELO Команда приветствия от SMTP-клиента
MAIL Идентифицирует почтовый адрес отправителя
RCPT Идентифицирует получателя
DATA Указывает на начало сообщения
SEND Отправляет письмо в терминал
SOML Send-or-Mail; отправляет письмо в почтовый ящик либо в терминал получателя
SAML Send-and-Mail; отправляет письмо и в почтовый ящик, и в терминал получателя
RSET Reset; отменяет соединение и всю переданную информацию
VRFY Удостоверяется, что такой пользователю есть на сервере
EXPN Удостоверяется, что такой адрес рассылки есть на сервере
HELP выводит список команд
NOOP No operation; нужен для траблшутинга, сервер отвечает OK
QUIT Заканчивает SMTP-сессию
TURN Запрос на изменение текущих ролей в SNMP-взаимодействии
```

Рассмотрим некоторые команды более детально.

HELO — как уже говорилось, протокол SMTP ограничивает длину команд четырьмя символами, поэтому оригинальное HELLO было урезано до HELO. Формат команды:

```
HELO hostname
```

Её цель — идентификация SMTP-клиента перед SMTP-сервером. К сожалению, сам протокол SMTP и метод идентификации был придуман до того, как появились злоумышленники и нежелательная рассылка, поэтому в HELO-сообщении клиент может написать любой домен, какой хочет. Поэтому большинство SMTP-серверов не обращают внимания на то, что указано в HELO-команде. Если SMTP-сервер действительно хочет знать, за какой домен отвечает отправитель, он, зная IP-адрес отправителя, обращается в DNS и спрашивает PTR для этого IP-адреса.
Для повышения безопасности и борьбы со спамом большинство SMTP-серверов не примут почту от SMTP-клиента, если нет валидной PTR-записи для IP-адреса.

Отправляя HELO-команду, клиент обозначает, что хочет начать новую SMTP-сессию с SMTP-сервером. Сервер же, отвечая на эту команду, подтверждает, что новая SMTP-сессия может быть установлена и он ожидает команд от клиента.

Клиент — это сервер или пользователь?

При использовании протокола SMTP вы должны различать действия пользователя и хоста. Когда пользователь создает письмо в MUA и нажимает кнопку «Отправить», его хост начинает взаимодействовать при помощи протокола SMTP с сервером получения почты. То есть для SMTP-протокола фактически нет разницы, получать/передавать почту от пользователя или делать это между MTA-серверами. Тот, кто в SMTP-сессии хочет отправить письмо, — SMTP-клиент, а тот, кто это письмо получает — SMTP-сервер. При этом оба участника могут быть MTA-серверами.

MAIL — команда начала передачи почты. Клиент указывает, кто является отправителем. Формат команды:

```
MAIL reverse-path
```

Reverse-path — не только идентификация отправителя, но и пояснение, как с ним связаться. Если отправитель — пользователь, формат команды будет примерно такой:

```
MAIL FROM:dradchuk@nodrop.in
```

Если письмо было отмаршрутизировано несколькими MTA между изначальным отправителем и конечным получателем, то каждый из них будет добавлять свою информацию в reverse-path. Так можно посмотреть, через какие почтовые сервера письмо прошло, что необходимо для траблшутинга и идентификации спама.

RCPT — команда, идентифицирующая получателя сообщения. Если получателей несколько, то каждый из них будет указан в отдельном RCPT-сообщении. Формат команды:

```
RCPT forward-path
```

Forward-path определяет конечного получателя сообщения. Обычно это полный адрес почты получателя, иногда это имя пользователя, если получатель локален для сервера.

```
RCPT TO:lliopt@yandex.ru
```

Либо, в случае, если идет локальная доставка почты:

```
RCPT TO:alice
```

Рассмотрим пример, когда SMTP-сервер **abc.example.com** получает следующее сообщение:

```
RCPT TO:user@zxc.example.com
```

Как видно, получатель не является пользователем домена abc.example.com, так как почта отправлена на zxc.example.com. Следовательно, SMTP-сервер должен понять, что ему делать дальше.

Сервер abc может принять почту и передать ее дальше MTA-получателю.

Сервер abc может отказаться принимать такое сообщение. В ответе он может указать, что SMTP-клиенту следует обратиться к почтовому серверу домена получателя. В дальнейшем SMTP-клиент может обратиться к правильному серверу для передачи почты.

Сервер abc может отказаться от передачи почты и сообщить SMTP-клиенту, что пересылка почты (smtp relay) запрещена на этом сервере.
DATA начинает передачу самого тела письма. Формат команды:

```
DATA
```

Всё, что следует после команды DATA, — часть сообщения. Обычно SMTP-сервер добавляет отметку о времени (timestamp) и детальную информацию return-path в начало сообщения.

Клиент отмечает конец сообщения отправкой точки:

```
<CR><LF>.<CR><LF>
```

Когда SMTP-сервер получает данную последовательность, он понимает, что передача письма окончена и он должен отправить клиенту код ответа.

Extended SMTP (ESMTP)

Спустя некоторое время использования SMTP стали понятны его ограничения, и в него были добавлены некоторые расширения. Так появился ESMTP.

ESMTP описан изанчально в RFC 1869 https://tools.ietf.org/html/rfc1869, но сейчас включен в уже указанный RFC 5321, объединяя SMTP и ESMTP.

В ESMTP команда приветствия HELO была заменена на новую — EHLO. Когда SMTP сервер получает команду EHLO, он понимает, что клиент поддерживает ESMTP и дополнительный список команд:

```
$ telnet mx.yandex.ru 25
Trying 77.88.21.89...
Connected to mx.yandex.ru.
Escape character is '^]'.
220 mxfront5g.mail.yandex.net (Want to use Yandex.Mail for your domain? Visit
http://pdd.yandex.ru)
EHLO nodrop.in
250-mxfront5g.mail.yandex.net
250-8BITMIME
250-PIPELINING
250-SIZE 42991616
250-STARTTLS
250-DSN
250 ENHANCEDSTATUSCODES
```

Local Mail Transfer Protocol (LMTP)

EMTP и ESMTP — протоколы, придуманные для передачи почты между удалёнными друг от друга SMTP-клиентом и SMTP-сервером.

Для локальной доставки почты в пределах одного сервера был придуман протокол LMTP.

Несмотря на то, что SMTP-клиент может указать несколько адресов RCPT TO, в тот момент, когда сообщение начало передаваться при помощи DATA-команды, SMTP-сервер отвечает одним-единственным кодом ответа. Если код ответа означает ошибку, SMTP-клиента не поймёт, кому письмо было доставлено, а кому — нет. Из-за этого SMTP-клиенту необходимо хранить письмо в очереди для следующей попытки передачи.

Когда письмо должно быть доставлено локальным MDA, не всегда возможно или необходимо хранить его в очереди. Вместо этого отправитель хотел бы иметь механизм получения информации о том, было ли принято или отклонено письмо для каждого получателя. Это позволит MTA и MDA избежать ненужного хранения писем в очереди, или даже иметь несколько MDA, обслуживающих один MTA.

LMTP был придуман, чтобы иметь возможность получать несколько кодов ответа внутри одной SMTP-сессии и избежать ненужной пересылки сообщений. Это делает протокол LMTP идеальным для взаимодействия между MTA и MDA.

LMTP использует тот же самый набор команд, что и SMTP/ESMTP, но с двумя дополнениями:

1. HELO/EHLO-команды заменены на LHLO. Это позволяет клиенту указать, что он поддерживает LMTP.
2. SMTP-код ответа создается на каждую команду RCPT TO.

```
$ telnet example.com 25
220 example.com LMTP server ready (smtpfeed 1.10, PID 24611)
LHLO test.net
220-mail.example.com Hello test.net
220-8BITMIME
220-DSN
220 SIZE
MAIL FROM: <a@test.net>
250 <a@test.net> Sender ok
RCPT TO: <a@example.com>
250 <a@example.com> Recipient ok
RCPT TO: <b@example.com>
250 <b@example.com> Recipient ok
RCPT TO: <c@example.com>
250 <c@example.com> Recipient ok
DATA
354 Enter mail, end with "." on a line by itself
From : a@test.net
To: a@example.com, b@example.com, c@example.com
Date: May 30, 2019 19:20:00
Subject: Test LMTP message

This is a test LMTP message.
.
250 <a@example.com>... OK (mail.example.com [192.168.1.10])
250 <b@example.com>... OK (mail.example.com [192.168.1.10])
550 <c@example.com>... User unable to accept messages
QUIT
221 mail.example.com closing connection
```

Как вы можете видеть в примере, первая LMTP-команда LHLO объясняет серверу, что клиент использует протокол LMTP, далее происходит стандартный для SMTP обмен командами. После того, как DATA-передача была закончена отправкой точки, несколько кодов ответа ушло клиенту. Два сообщили, что письма были переданы пользователям, а последнее не было доставлено. Теперь LMTP-клиент может известить отправителя о неуспешной попытке доставки сообщения.

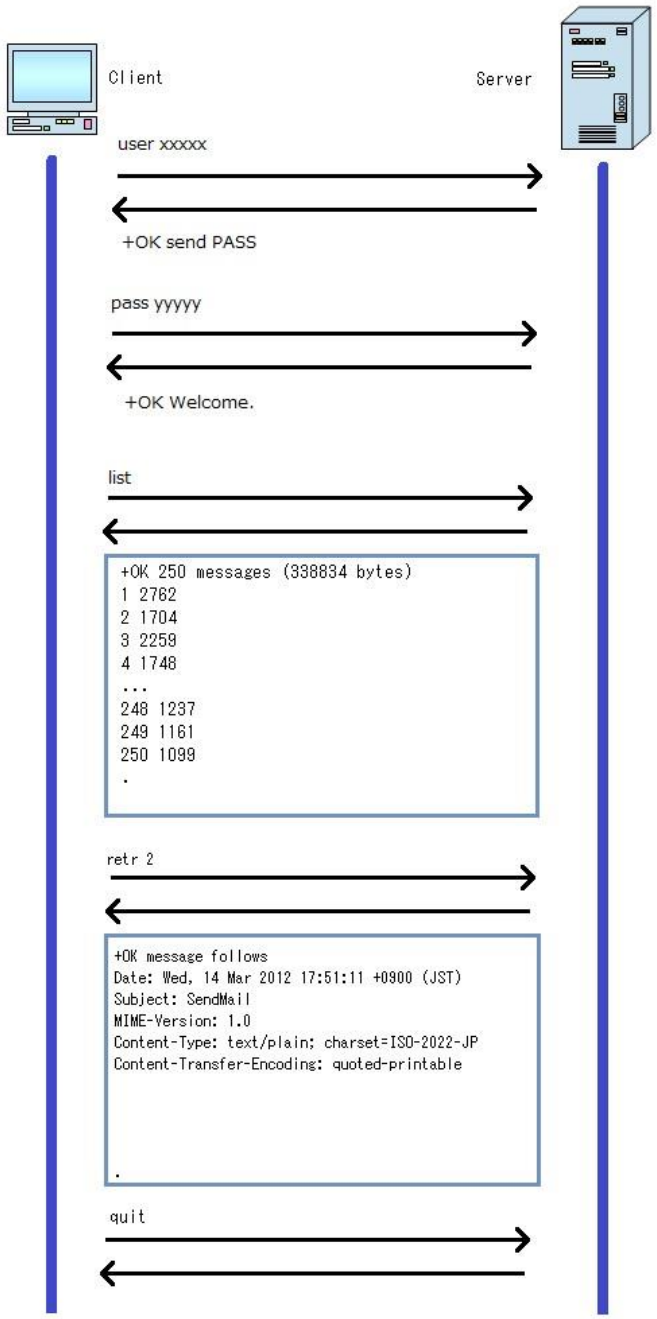
MUA-протоколы

Протоколы MUA позволяют пользователям читать почту из их почтовых ящиков. Передача почты локальным пользователем операционной системы, где установлен почтовый сервер, не проблема, так как её можно передать в терминал. Но доставка почты большому количеству не локальных пользователей — уже нетривиальная задача.

Для решения этой проблемы есть несколько протоколов взаимодействия удаленных пользователей с их почтовыми ящиками. Каждый пользователь имеет возможность при помощи MUA подключиться к своему почтовому ящику для работы с письмами.

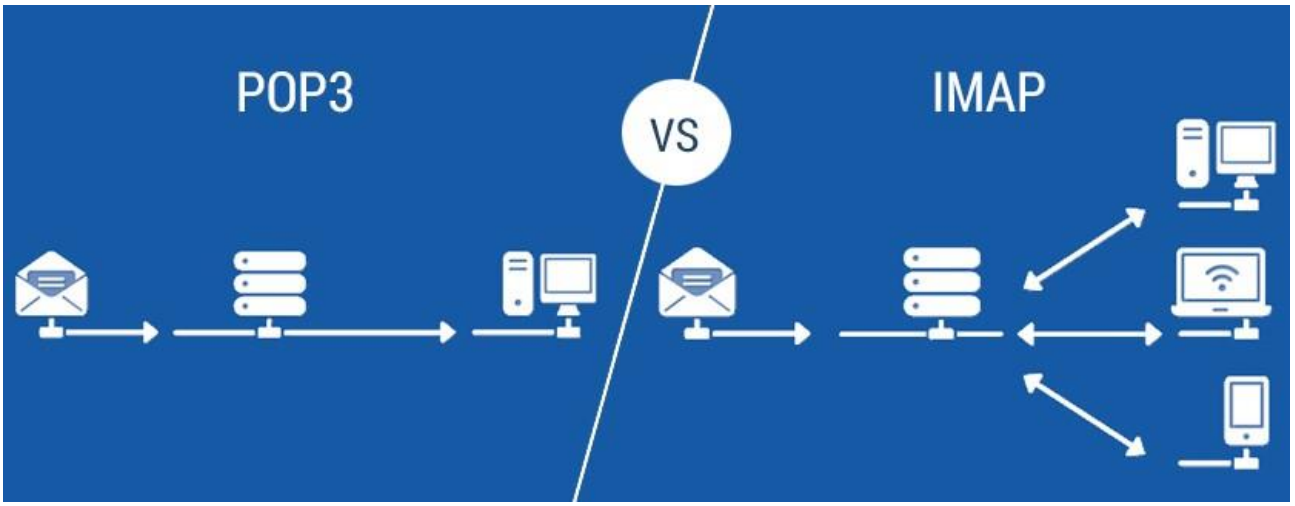
Post Office Protocol (POP)

Простейший протокол взаимодействия с почтовым ящиком. При использовании POP все письма из почтового ящика пользователя передаются на его локальный компьютер. Обычно после этого MUA удаляет все письма с сервера, оставляя лишь их локальные копии.



Interactive Mail Access Protocol (IMAP)

Протокол IMAP позволяет компьютеру пользователя удалённо управлять письмами в папках, которые находятся на сервере. При подключении по IMAP письмо остаётся на почтовом сервере и не скачивается лишь для того, чтобы показать заголовок письма пользователю. IMAP — достаточно удобный протокол, если вы пользуетесь несколькими почтовыми клиентами и устройствами для чтения почты, так как копия письма всегда хранится на сервере.



Факторы, влияющие на доставку почты

Так как сам по себе протокол SMTP не имеет никаких встроенных методов борьбы со спамом, MTA-серверы основываются на DNS, чтобы понять легитимность отправителя.

При получении письма почтовый сервер-получатель проводит ряд проверок. Чем больше проверок будет пройдено с положительным результатом, тем меньше вероятность того, что письмо попадет в спам.

Pointer (PTR)

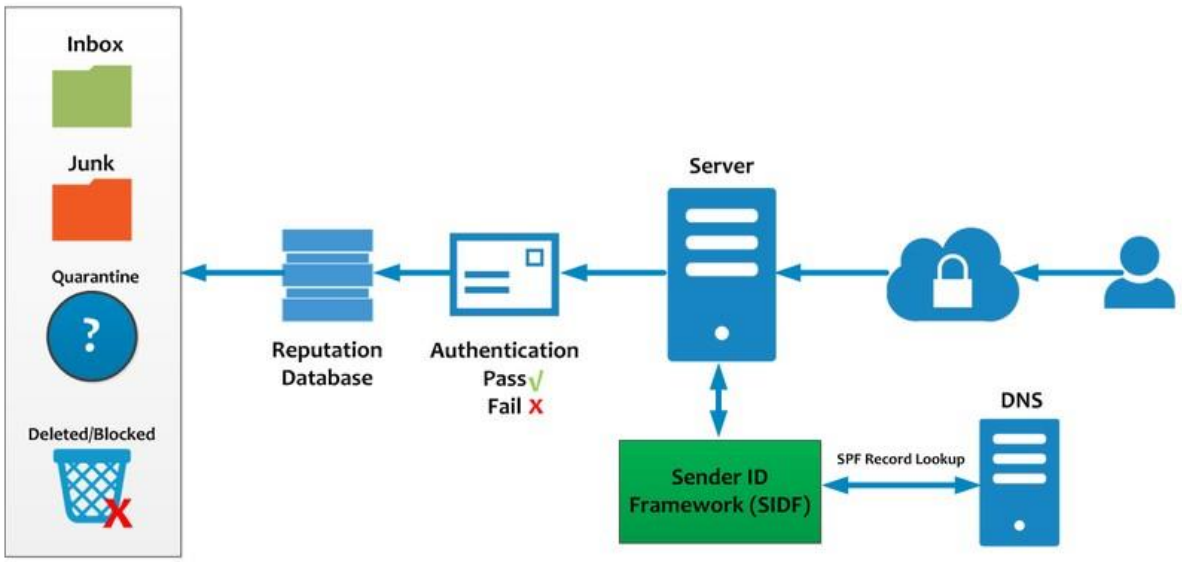
PTR-запись — обратная запись, которая связывает IP-адрес с именем домена и отвечает за его преобразование в доменное имя.

В целях борьбы со спамом многие MTA-серверы проверяют наличие PTR-записи для хоста, от которого происходит отправка, и если видят несоответствие с именем, указанным в HELO/EHLO, с высокой вероятностью пометят это письмо как спам. То есть IP-адрес имени хоста (хост этим именем представляется), который постучался к серверу-получателю на почтовый порт для передачи письма, должен преобразоваться в это имя.

Sender Policy Framework (SPF)

Sender Policy Framework — расширение для протокола отправки электронной почты через SMTP. SPF описан в RFC 7208.

SPF позволяет владельцу домена указать в TXT-записи домена специальным образом сформированную строку, указывающую на список серверов, имеющих право отправлять почту с обратными адресами в этом домене. То есть в этой записи мы сообщаем, с каких почтовых серверов можно посылать письма от имени нашего домена.



В SPF-записях можно использовать следующие опции:

- v=spf1 — версия, всегда spf1,
- a — разрешает отправлять письма с адреса, который указан в A- или AAAA-записи домена отправителя,
- mx — разрешает отправлять письма с адреса, который указан в MX-записи домена (для A и MX можно указать и другой домен, например, при значении a:example.com будет разрешена A-запись не домена отправителя, а example.com),
- -all — диктует, что будет происходить с письмами, которые не соответствуют политике, то есть не совпадают с тем, что указано в записи:
  - - — отклонять,
  - + — пропускать,
  - ~ — принимать, но, возможно, пометить как спам/дополнительно проверить, ○ ? — нейтрально.
- include включает в себя хосты, разрешенные SPF-записью указанного домена,
- ptr проверяет PTR-запись IP-адреса отправителя. Если она сходится с указанным доменом, то механизм проверки выдает положительный результат. Например, разрешено отправлять всем IP-адресам, PTR-запись которых направлены на указанный домен.
- exists — выполняется проверка, резолвится ли домен на какой-либо IP-адрес. То есть выполняется проверка работоспособности доменного имени.
- redirect указывает получателю, что нужно проверять SPF-запись указанного домена вместо текущего.

Простой пример SPF-записи:

```
$ dig yandex.ru TXT | grep spf
yandex.ru.              1185    IN      TXT     "v=spf1 redirect=_spf.yandex.ru"
```

Тут мы видим, что домен yandex.ru перенаправляет на другой — _spf.yandex.ru для проверки spf.

```
$ dig _spf.yandex.ru TXT | grep spf
```

```
_spf.yandex.ru.                    3600        IN            TXT          "v=spf1
 include:_spf-ipv4.yandex.ru include:_spf-ipv6.yandex.ru ~all"
```

include:_spf-ipv4.yandex.ru показывает доменное имя, за которым скрывается список ipv4-адресов, с которых разрешена отправка почты из домена yandex.ru.

```
$ dig _spf-ipv4.yandex.ru  TXT | grep spf
_spf-ipv4.yandex.ru.    3600    IN    TXT    "v=spf1    ip4:213.180.223.192/26
ip4:37.9.109.0/24   ip4:37.140.128.0/18   ip4:5.45.192.0/19   ip4:5.255.192.0/18
ip4:77.88.0.0/18  ip4:87.250.224.0/19  ip4:93.158.136.48/28  ip4:95.108.130.0/23
ip4:95.108.192.0/18 ip4:141.8.132.0/24 ~all"
```

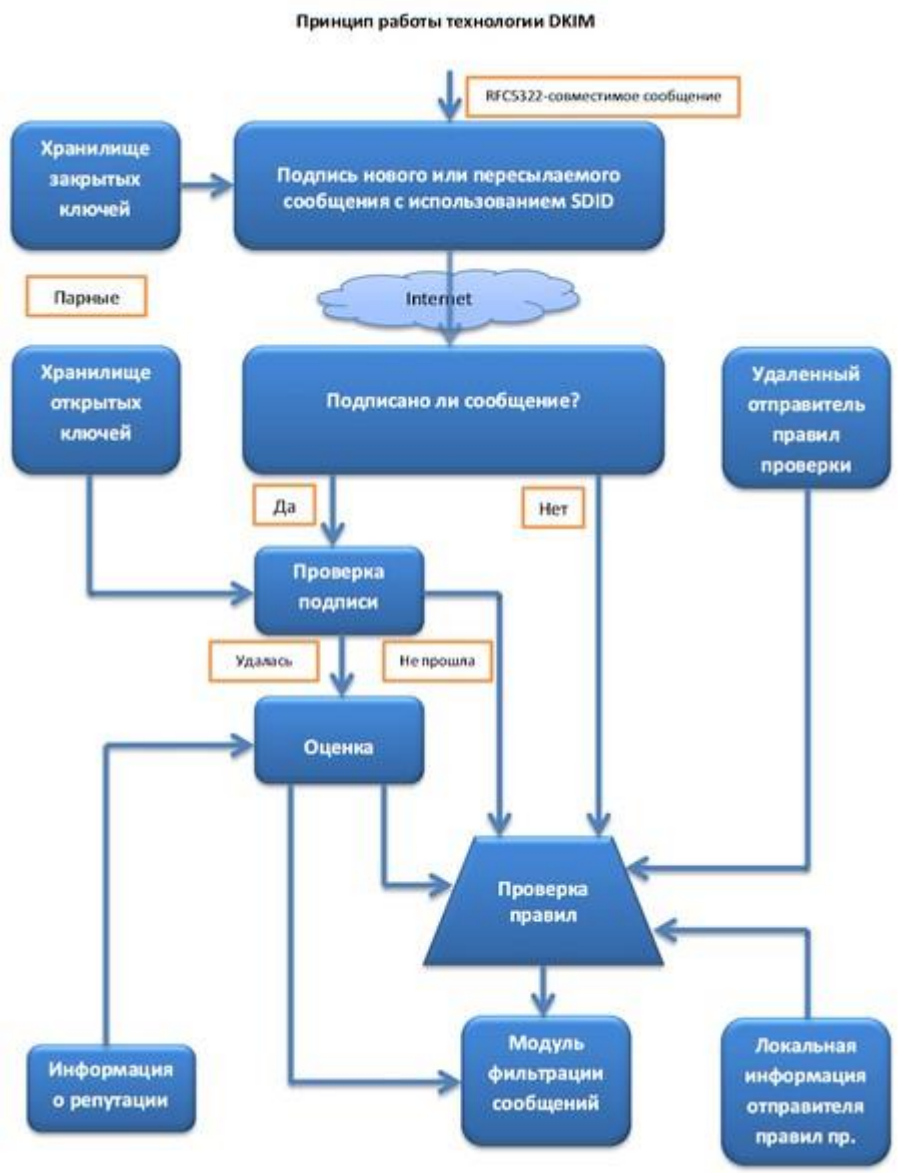
ip4:213.180.223.192/26 — пример подсети, из которой разрешена отправка почты.

~all говорит, что письма из домена yandex.ru могут быть отправлены перечисленными IP-адресами, но могут быть еще какие-то отправители, которые не указаны в этом списке.

```
$ dig _spf-ipv4.yandex.ru  TXT | grep spf
_spf-ipv4.yandex.ru.    3600    IN    TXT    "v=spf1    ip4:213.180.223.192/26
ip4:37.9.109.0/24   ip4:37.140.128.0/18   ip4:5.45.192.0/19   ip4:5.255.192.0/18
ip4:77.88.0.0/18  ip4:87.250.224.0/19  ip4:93.158.136.48/28  ip4:95.108.130.0/23
ip4:95.108.192.0/18 ip4:141.8.132.0/24 ~all"
```

DomainKeys Identified Mail

DKIM необходим для определения достоверности отправителя письма путём добавления цифровой подписи, которую можно проверить открытым ключом шифрования, указанным в текстовой записи домена, который хранится в TXT-записи домена.



Для работы с DKIM нужно выполнение следующих пунктов:

- Поддержка DKIM почтовым сервером для подписывания отправляемой почты;
- Создание приватного и публичного ключа шифрования;
- Занесение в DNS домена записей, связанных с DKIM, требует специального имени — _domainkey, например:

```
<имя>._domainkey.example.com. TXT "v=DKIM1; k=rsa; t=s; p=<публичный ключ>"
```

- <имя> — так называемый domain selector, которых может быть несколько, например для каждого почтового сервера, если их у Вас несколько.
- v — версия DKIM, всегда принимает значение v=DKIM1 (обязательный аргумент);
- k — тип ключа, всегда k=rsa (по крайней мере, на текущий момент); ● p — публичный ключ, кодированный в base64 (обязательный аргумент); ● t — флаги:
    - t=y — режим тестирования, отличаются от неподписанных и нужны лишь для отслеживания результатов;
    - t=s — означает, что запись будет использована только для домена, к которому относится запись; не рекомендуется, если используются субдомены;
    - h — предпочитаемый hash-алгоритм, может принимать значения h=sha1 и h=sha256;

○ s — тип сервиса, использующего DKIM. Принимает значения s=email (электронная почта) и s=* (все сервисы) По умолчанию ". ○ ; — разделитель.

```
$ dig mail._domainkey.yandex.ru  TXT | grep DKIM
mail._domainkey.yandex.ru. 3592 IN           TXT         "v=DKIM1; g=*; k=rsa;
p=MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDEc6Lkc9kLHjIxLkeszz1dYzGIfPH8qaUx2wLo
jY
efUzZiCjyl0s/YT17WJMfGFZkl0gHgkEj5/I2C72MmaHVtTFzNqD48ZuqVydlDyfLed0A6vxb+MS34
DI bpCgCi0HxQO1QRG7PechKza0iazWTIAQ1xRU24ZYM70kGDzhFHSwIDAQAB"
```

Кроме этого, существует необязательная запись, которая говорит, что делать с неподписанными письмами:

```
_adsp._domainkey.example.com. TXT "dkim=all"
```

dkim= может принимать следующие значения:

- all — отправка неподписанных сообщений запрещена,
- discardable — все неподписанные сообщения должны быть заблокированы на стороне получателя,
- unknown — отправка неподписанных сообщений разрешена (значение по умолчанию).

```
$ dig _adsp._domainkey.yandex.ru  TXT | grep -i dkim

_adsp._domainkey.yandex.ru. 3499 IN     TXT     "dkim=unknown"
```

Domain-based Message Authentication, Reporting and Conformance (DMARC)

DMARC — это техническая спецификация, предназначенная для снижения количества спама и фишинговых электронных писем, основанная на идентификации почтовых доменов отправителя на основании правил и признаков, заданных на почтовом сервере получателя. Почтовый сервер сам решает, хорошее это сообщение или плохое, и действует согласно DMARC-записи в DNS. Письма считаются поддельными, если они не проходят проверку SPF и DKIM, поэтому dmarc нужно прописывать только после их настройки. Также можно получать отчеты от почтовых систем, которые сообщают, сколько через них пыталось пройти или прошло поддельных писем на ваш домен.

Типичная запись выглядит так:

```
_dmarc.example.com TXT "v=DMARC1; p=none; rua=mailto:postmaster@example.com"
```

- v — версия, принимает значение v=DMARC1,
- p — правило для домена, может принимать значения:
  - ○ none — не делает ничего, кроме подготовки отчетов, ○ quarantine — добавляет письмо в спам, ○ reject — отклоняет письмо.
- sp отвечает за субдомены и принимает такие же значения, как и p,

- aspf и adkim позволяют проверять соответствия записей и могут принимать значения r и s, где r — relaxed — более мягкая проверка, чем s — strict,
- pct отвечает за количество писем, подлежащих фильтрации, указывается в процентах, например, pct=20 будет фильтровать 20% писем,
- rua позволяет отправлять ежедневные отчеты на email, пример: rua=mailto:postmaster@example.com, также можно указать несколько email через пробел (rua=mailto:postmaster@example.com mailto:dmarc@example.com),
- ruf — отчеты писем, не прошедшие проверку DMARC. В остальном все так же, как и выше.

```
$ dig _dmarc.yandex.ru  TXT | grep DMAC

_dmarc.yandex.ru.          3594      IN        TXT       "v=DMARC1; p=none; fo=1;
rua=mailto:dmarc_agg@auth.returnpath.net,mailto:dmarc-rua@yandex.ru;
ruf=mailto:dmarc_afrf@auth.returnpath.net"
```

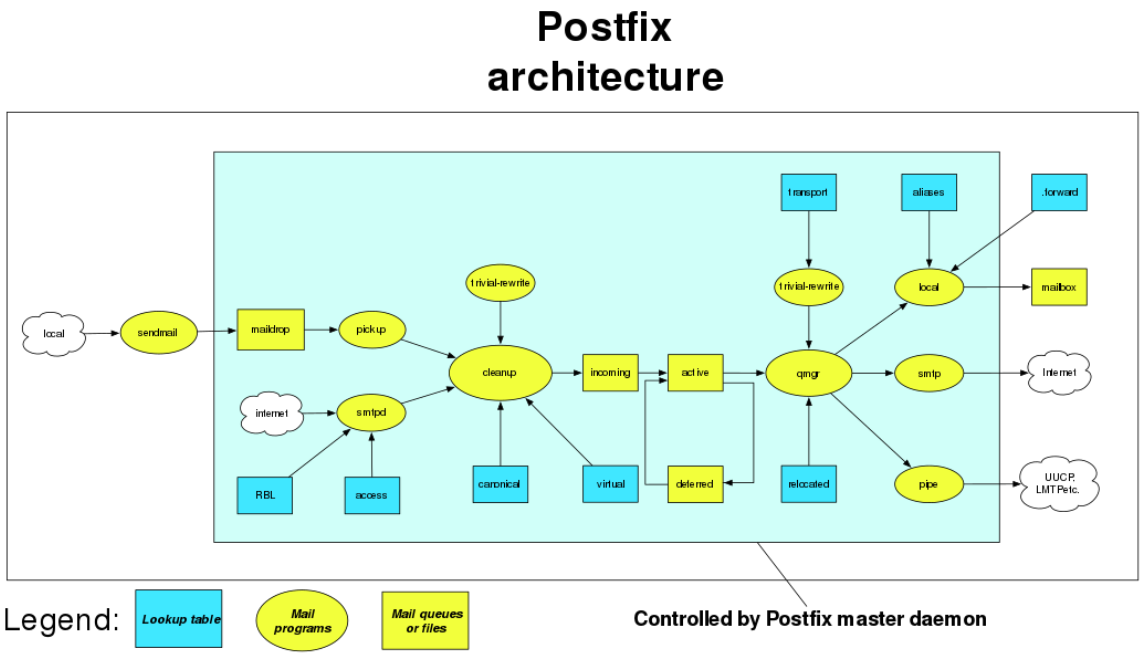
Настройка почты

В этой части мы рассмотрим настройку почтового сервера, который будет включать в себя MTA, MDA, а также веб-интерфейс для взаимодействия с почтовым ящиком при помощи браузера.

Postfix

Postfix — Mail Transfer Agent, который может маршрутизировать и доставлять электронную почту. Так как Postfix является MTA, он не предоставляет сервиса по забору почты для конечных пользователей, то есть не поддерживает протоколов POP, IMAP. Подробности — в текущей документацией по Postfix.

Postfix представляет собой несколько отдельных сервисов, которые взаимодействуют друг с другом. Каждый демон имеет свою зону ответственности и не зависим от других, при этом, все они контролируются материнским процессом master.



Получение почты

Новые письма могут быть доставлены в Postfix несколькими методами. Первый и самый популярный — при помощи протокола SMTP и соответствующего демона smtpd. Кроме этого, есть возможность получить письмо локально, используя sendmail. Это стандартный вариант отправки почты скриптами и программами, запущенными на хосте. Множество *NIX MUA, например Mutt, Pine и другие используют интерфейс sendmail, чтобы отправлять новые сообщения. Sendmail передает письмо компоненту postdrop, который, в свою очередь, помещает письмо в maildrop-директорию, находящуюся внутри почтовой очереди Postfix. Далее pickup-демон ожидает наличия письма в maildrop-директории и, если оно там, передает его cleanup-демону. С этого момента письма, отправленные через sendmail, обрабатываются так же, как и те, что пришли по SMTP-протоколу.

Когда smtpd или pickup получает новое письмо, он передает его следующему в очереди демону — cleanup. Cleanup применяет ограничения, настроенные на сервере, например, проверку на превышение максимального размера письма или проверку контента, добавляет заголовки, если требуется, и так далее. Cleanup использует trivial-rewrite-демона для перезаписи почтовых адресов. Закончив работу с письмом, cleanup помещает письмо во входящую очередь и оповещает queue manager.

Планировка доставки почты

Queue manager (qmgr) — сервис, ответственный за планировку доставки почты. Чтобы понять, как и кому письмо должно быть доставлено, qmgr использует trivial-rewrite. Qmgr запрашивает доставку почты у master-демона, а также обрабатывает результаты доставки почты.

Queue manager отвечает за все сообщения с момента, когда демон cleanup передает их для обработки, пока письма не будут удалены из очереди. Причин удаления из очереди две — письмо успешно доставлено, либо находилось в очереди так долго, что postfix решил, что оно не доставлено. О недоставке письма отправителю сообщает демон bounce.

Queue manager использует несколько каталогов для разных целей. Входящая очередь отслеживается на наличие новых писем, за ней следует активная очередь. Активная очередь содержит письма, которые готовы к доставке получателю. Если попытка доставки не удалась, письмо перемещается в отложенную очередь. Эта очередь сканируется время от времени. Спустя некоторое время файл этого письма будет снова перемещен в активную очередь. Повторная попытка доставки почты при сканировании очереди зависит от двух факторов: время с момента получения письма и параметры конфигурации, которые устанавливают минимальный и максимальный интервал между повторными попытками.

Доставка почты

Postfix обладает несколькими агентами, которые применяются для доставки почты. Агенты доставки почты — последние демоны, которые обрабатывают почту перед тем, как она покинет сервер.

Postfix-SMTP-клиент применяется для доставки почты при помощи протокола SMTP. LMTP-клиент (lmtp) применяется, соответственно, для доставки почты через LMTP-протокол.

Local delivery agent (local) доставляет почту пользователям с обычными аккаунтами (рядовые пользователи, имеющие доступ к shell). Почту для виртуальных пользователей (virtual mailbox users) доставляет компонент virtual.

Настройка Postfix

Как и большинство Linux-приложений, Postfix узнаёт о своей конфигурации из текстового файла, находящегося в папке /etc/postfix. Для настройки используются 2 файла — master.cf и main.cf, а также вспомогательные файлы, которые вы сможете создать, если потребуется.

Для применения любых изменений сервис Postix должен быть перезапущен.

```
user@testserver$ sudo systemctl restart postfix
```

Main.cf

Файл, модифицировать который вы будете наиболее часто, так как именно он отвечает за настройку и поведение демона Postfix. Конфигурацию в целом можно описать так:

```
parameter = value
```

Если параметр не указан явно в файле конфигурации, будет использовано значение по умолчанию. В случае, если один и тот же параметр указан более одного раза, будет использовано последнее значение. При этом очередность внутри первой строки имеет значение, например, данные параметры не обязательно эквиваленты:

```
parameter = value1, value2

parameter = value2, value1
```

Если строка начинается с пробела, это означает, что она продолжает предыдущую. Например, эти строки эквивалентны друг другу:

```
smtpd_recipient_restrictions = permit_mynetworks, reject_unauth_destination

smtpd_recipient_restrictions =

        permit_mynetworks,
        reject_unauth_destination
```

Давайте рассмотрим основные настройки main.cf. Для начала мы должны сообщить Postfix, за какой домен он отвечает при помощи параметра mydomain. Например, если наш домен http://www.example.com, устанавливаем:

```
mydomain = example.com
```

Значение в mydomain определяет то, как Postfix трансформирует имена хостов, которые указаны не так. Хостнеймы будут автоматом модифицированы до FQDN при помощи добавления mydomain, например, test будет преобразован в test.example.com.

Соответственно, есть параметр hostname, который отвечает за имя хоста и явно говорит Postfix, какое имя использовать:

```
hostname = test
```

Hostname, как и некоторые другие параметры, применяются, когда сервер Postfix SMTP приветствует клиента и тот отправляет HELO серверу. Обычно Postfix может получить значение hostname, исходя из заданного серверу имени хоста, но иногда требуется явно указать имя.

Следующий параметр связан с mydomain и называется myorigin. Этот параметр определяет домен, который должен быть использован для определения доменной части почтового адреса, если она не указана. Это может казаться нерелевантным, но, например, письма, отправленные при помощи sendmail, будут иметь текущее имя пользователя как адрес отправителя, и, так как имена пользователей не содержат доменной части, к юзернейму будет добавлено значение параметра myorigin перед тем как письмо будет куда-либо доставлено. Настройка по умолчанию:

```
myorigin = $myhostname
```

Параметр mydestination также достаточно важен - он говорит Postfix какие домены являются локальными, то есть письма должны быть доставлены локально UNIX-пользователям этого сервера.

```
mydestination = $mydomain, $myhostname, example.com, localhost.$mydomain
```

В отличие от параметров mydomain и myorigin, значение mydestination может содержать несколько доменов, разделённых пробелом или запятой. Например, указав example.com, Postfix-сервер примет сообщение на адрес user1@example.com и доставит его UNIX-пользователю user1.

Следует понимать, что все домены, перечисленные в mydestination, равны с точки зрения Postfix. То есть, если в списке присутствуют домены example.com и example.net, то почта user1@example.com будет равна почте user1@example.net. Если необходимо иметь уникальных пользователей для разных доменов, можно использовать функцию Virtual Alias Domain.

Достаточно важные параметры — mynetworks и mynetworks_style, которые контролируют каким хостам можно использовать этот SMTP-сервер как релей. Установка неверных значений может привести к тому, что ваш сервер будет использован в качестве почтового релея в спам-рассылках. Значения по умолчанию ограничивают использование сервера только для подсетей, назначенных на его интерфейсы.

В некоторых случаях интернет-провайдеры блокируют возможность использования почтовых серверов напрямую через SMTP-порт (TCP-порт 25). Вместо этого провайдер предоставляет адрес почтового релея, через который должны проходить все исходящие сообщения.

В таком случае вам следует настроить Postfix на доставку исходящей почты не напрямую, а через релей при помощи параметра relayhost:

```
relayhost = example.com

relayhost = [mail.example.com]

relayhost = [192.0.2.1]
```

В первом случае, если запись указана без [], Postfix будет осуществлять поиск MX-записи для домена, ровно так же, как он делал бы это в случае с прямой пересылкой почты. Если же мы указываем имя релея в квадратных скобках, поиск MX-записи не осуществляется. Квадратные скобки также необходимы при использовании IP-адреса релея вместо доменного имени.

Параметр inet_interfaces описывает, какие сетевые интерфейсы будут использованы для ожидания новых подключений, либо для установки соединения при отправке почты. Если у вас несколько интерфейсов и вы не хотите, чтобы Postfix использовал их все, вы можете перечислить их имена.

Некоторые дистрибутивы ограничивают inet_interfaces значением localhost, что означает, что Postfix будет ожидать новых подключений только локально, на интерфейсе loopback, что является нерабочей конфигураций для SMTP-сервера.

Master.cf

Master.cf отвечает за настройку master-демона, который был описан ранее. Каждая строка этого файла отвечает за настройку определенного компонента. Например, smtpd — демон, отвечающий за SMTP-соединения, local — за доставку локальным юзерам и так далее.

Обычно master.cf остаётся неизменным для большинства простых инсталляций Postfix.

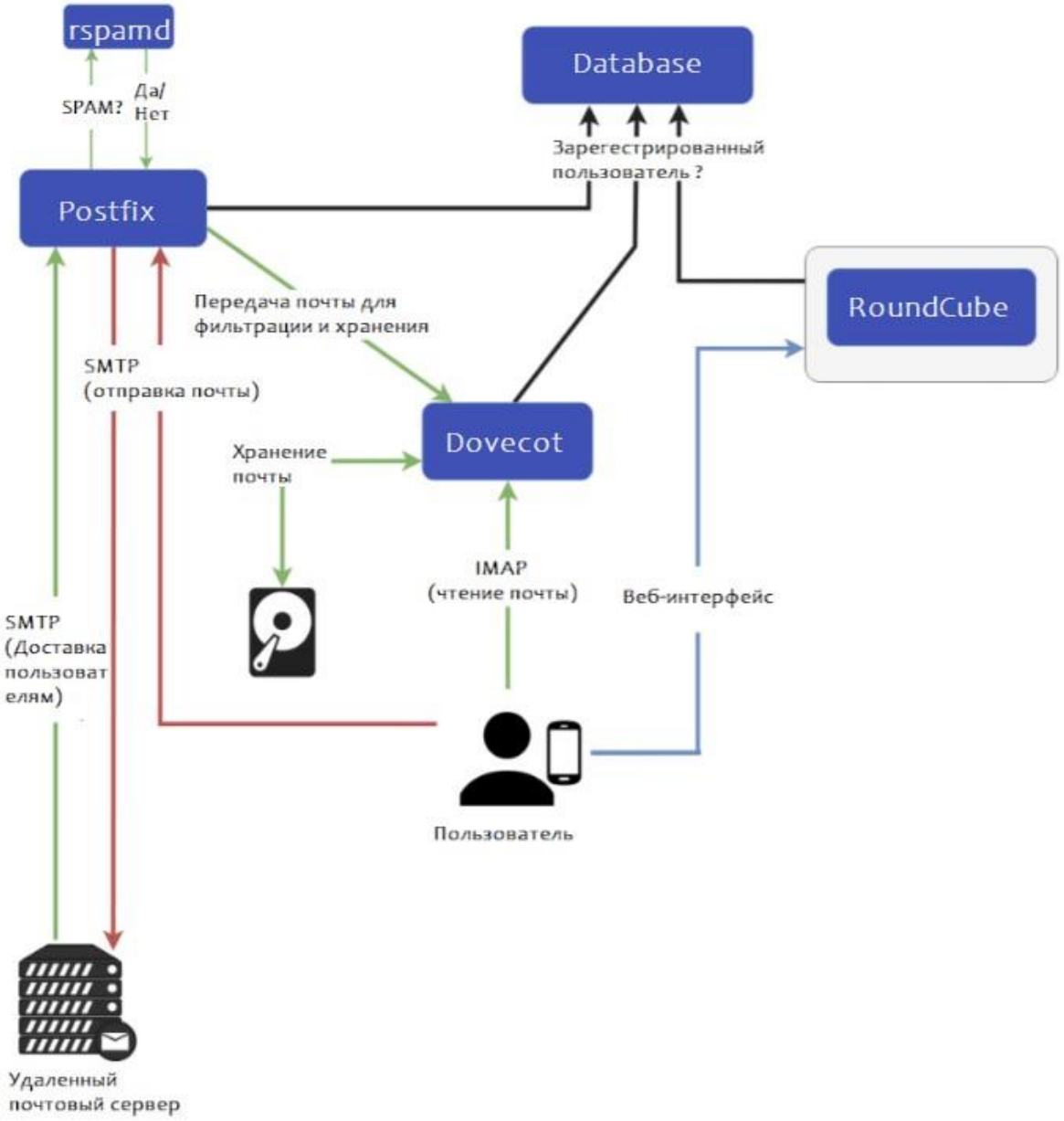
Mailcow

Mailcow — контейнезированный почтовый сервер с графическим интерфейсом для удобства пользования. Контейнеры разворачиваются при помощи docker-compose и находятся в одной виртуальной коммутируемой сети на одной виртуальной машине. Кроме этого, можно запустить Mailcow в Kubernetes.

Mailcow состоит из следующих компонентов:

- Postfix.
- Dovecot.
- ClamAV (опционально).
- Solr (опционально).
- Memcached.
- Redis.
- MySQL.
- Unbound (как резолвер).
- PHP-FPM.
- ACME-Client.
- Nginx.
- Rspamd.
- SOGo.
- Netfilter.
- Watchdog.

Общая логика работы почтового сервера выглядит так:



Так как сервис контейнеризирован, требования к системе в основном касаются доступных ресурсов:

- CPU — 1 GHz.
- RAM — 2GB.
- Storage — 15G.

В нашем случае Mailcow будет установлен на CentOS 7.6 Core, поэтому сначала подготовим систему.

Подготовка CentOS 7.6 core

Настройка сервера

Меняем имя сервера на соответствующее:

```
[centos@centos-basic-1-2-20gb ~]$ sudo hostnamectl set-hostname mail.nodrop.in
```

Обновляем систему и перезагружаем:

```
[root@mail ~]# yum update -y

[root@mail ~]# reboot
```

Устанавливаем набор утилит, которые нам потребуются в дальнейшем:

```
[root@mail centos]# yum install -y epel-release     yum-utils  unzip  curl wget
bash-completion    policycoreutils-python    mlocate    bzip2    vim    yum-
device-mapper-persistent-data lvm2 git                                  utils
```

Проверяем настройки времени на сервере, и, если требуется, настраиваем ntp:

```
[centos@mail ~]$ timedatectl status

      Local time: Sun 2019-06-09 13:28:43 UTC

  Universal time: Sun 2019-06-09 13:28:43 UTC

        RTC time: Sun 2019-06-09 13:28:43

       Time zone: UTC (UTC, +0000)

     NTP enabled: yes

NTP synchronized: yes
 RTC in local TZ: no
 DST active: n/a
```

Так как все сервисы у нас будут запущены в контейнерах, нет необходимости в стандартно запущеном Postfix. Также мы должны убедиться, что TCP-порты 25|80|110|143|443|465|587|993|995 не заняты приложениями.

```
[centos@mail     ~]$     sudo     netstat     -tulpan     |     grep     -E     -w
 '25|80|110|143|443|¦65|587|993|995'

tcp           0       0 127.0.0.1:25             0.0.0.0:*                LISTEN
 2927/master

tcp6         0       0 ::1:25                   :::*                     LISTEN
 2927/master
```

Как видите, в нашем случае уже есть сервис Postfix, который запущен и занял 25-й порт.
Следовательно, надо его отключить

```
[centos@mail ~]$ sudo systemctl stop postfix

[centos@mail ~]$ sudo systemctl disable postfix

Removed symlink /etc/systemd/system/multi-
user.target.wants/postfix.service.

[centos@mail    ~]$        sudo    netstat    -tulpan    |    grep    -E
'25|80|110|143|443|465|587|993|995'

[centos@mail ~]$                                                    -
                                                                    w
```

Порты распределены по следующим сервисам:

Service	Protocol	Port	Container	Variable
Postfix SMTP	TCP	25	postfix-mailcow	${SMTP_PORT}
Postfix SMTPS	TCP	465	postfix-mailcow	${SMTPS_PORT}
Postfix Submission	TCP	587	postfix-mailcow	${SUBMISSION_PORT}
Dovecot IMAP	TCP	143	dovecot-mailcow	${IMAP_PORT}
Dovecot IMAPS	TCP	993	dovecot-mailcow	${IMAPS_PORT}
Dovecot POP3	TCP	110	dovecot-mailcow	${POP_PORT}
Dovecot POP3S	TCP	995	dovecot-mailcow	${POPS_PORT}
Dovecot ManageSieve	TCP	4190	dovecot-mailcow	${SIEVE_PORT}
HTTP(S)	TCP	80/443	nginx-mailcow	${HTTP_PORT} / ${HTTPS_PORT}

Установка Docker

Добавляем docker-ce-репозиторий:

```
[root@mail centos]# yum-config-manager --add-
 https://download.docker.com/linux/centos/docker-ce.repo repo
```

Устанавливаем docker-ce:

```
[root@mail centos]# yum -y  install docker-ce
```

Добавляем пользователя (в нашем случае пользователя CentOS) в группу Docker и проверяем:

```
[centos@mail ~]$ sudo usermod -aG docker $(whoami)

[centos@mail ~]$ id centos

uid=1000(centos) gid=1000(centos)
groups=1000(centos),4(adm),10(wheel),190(systemd-journal),994(docker)
```

Включаем Docker при загрузке операционной системы:

```
[centos@mail ~]$ sudo systemctl enable docker.service

Created symlink from /etc/systemd/system/multi-user.target.wants/docker.service
to /usr/lib/systemd/system/docker.service.
```

Установка Docker Compose

Устанавливаем python-pip:

```
[centos@mail ~]$ sudo yum install -y python-pip
```

Устанавливаем Docker Compose:

```
[centos@mail ~]$ sudo pip install docker-compose
```

Обновляем pip^

```
[centos@mail ~]$ sudo pip install --upgrade pip
```

Проверяем Docker Compose:

```
[centos@mail ~]$ docker-compose version docker-

compose version 1.24.0, build 0aa5906 docker-py

version: 3.7.2
```

```
CPython version: 2.7.5

OpenSSL version: OpenSSL 1.0.2k-fips  26 Jan 2017
```

Настройка DNS

Для нормального функционирования почты, требуется настроить DNS. В нашем случае, godaddy.com при покупке домена предлагает сервис DNS, которым мы и пользуемся. Для PTR-записи нужно обратиться к владельцу IP-адреса, чтобы он на своём DNS для IP-адреса, который назначен нашему серверу, внес соответствующую запись с доменным именем.

Для начала посмотрим на настройки DNS у godaddy для нашего домена:

```
; SOA Record       IN                                dns.jomax.net. (
nodrop.in. 3600

                        SOA   ns21.domaincontrol.com.

                              2019052519

                              28800

                              7200

                              604800

                              3600

                              )

; NS Records

@       3600    IN    NS    ns21.domaincontrol.com.

@       3600    IN    NS    ns22.domaincontrol.com.

; MX Records

@       3600    IN    MX    10    mail.nodrop.in.

; A Records

@       3600    IN    A     89.208.87.64
```

```
mail 3600     IN   A     89.208.87.64

www    3600   IN   A     89.208.87.64

 ; CNAME Records

imap 3600     IN
                CNAME mail.nodrop.in.

smtp 3600     IN   CNAME mail.nodrop.in.

 ; TXT Records
```

```
@      3600   IN   TXT   "v=spf1 mx -all"

@      3600   IN   TXT
"v=DKIM1;k=rsa;t=s;s=email;p=MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAoD0i4x
K
I8YljUbD2ztqo+oixzjBgojm7WuecpCL1D/h23lxN4ePwgqbLhy42s1dDrZV+l3vMEJdcLiNpB20bcM
u
zVwvjD0UKUzd9+xIM4MSZJhvXdTFliW8W4foaWTomTW1U8eG9QR1ozB+1BdA2ZZB58+JnGdJH7b9Qtb
K
V/GUDWwVkazERKZHmzLDMAGlI6bRLx9d3/y6qyCTlF0PbLIgYz4p9UaUki+d3+o43wNxC+vEGj4+lCX
m
XDkMM2TSKoWj+/NfN5Id9eemZ7dUdKa5qEdZ87lghxhsD/oIL0+3b5rsWicOh8tNc9RNc8jxx4TjAxI
y AtHvXCPkGf/8v3QIDAQAB"

_dmarc3600 IN    TXT   ""v=DMARC1; aspf=s;     adkim=s;    pct=100;    p=reject;
rua=mailto:postmaster@nodrop.in;"

; SRV Records

_imap._tcp.@    3600      IN SRV     0 1 143   mail.nodrop.in.

_imaps._tcp.@   3600      IN SRV     0 1 993   mail.nodrop.in.

_pop3._tcp.@    3600     IN SRV     0 1 110   mail.nodrop.in.

_pop3s._tcp.@   3600    IN SRV     0 1 995   mail.nodrop.in.

_submission._tcp.@ 3600   IN SRV     0 1 587   mail.nodrop.in.

_smtps._tcp.@    3600     IN SRV     0 1 465   mail.nodrop.in.

_sieve._tcp.@    3600     IN SRV     0 1 4190  mail.nodrop.in.

_autodiscover._tcp.@  3600 IN SRV     0 1 443   mail.nodrop.in.

_carddavs._tcp.@  3600    IN SRV     0 1 443   mail.nodrop.in.

_caldavs._tcp.@   3600   IN SRV     0 1 443   mail.nodrop.in.

; CAA Records

@      3600   IN   CAA   0     issue "letsencrypt.org"
```

```
@       3600    IN    CAA    0      issuewild    ";"

@       3600    IN    CAA    0      iodef "mailto:postmaster@nodrop.in"
```

Как видно, файл зоны для домена nodrop.in живёт на двух DNS-серверах — ns21.domaincontrol.com и ns22.domaincontrol.com.

- Почтовый сервер, отвечающий за наш домен, имеет имя mail.nodrop.in, и он у нас всего один с приоритетом 10.
- Записи созданы только для имен nodrop.in, mail.nodrop.in и www.nodrop.in. Все они указывают на один и тот же IP-адрес: 89.208.87.64, так как сервер у нас один.
- Кроме этого, у нас создано несколько алиасов, smpt.nodrop.in и imap.nodrop.in, которые указывают на имя mail.nodrop.in.
- SPF-запись разрешает отправку почты только с MX-серверов ("v=spf1 mx -all). ● DKIM-запись включает в себя публичный RSA-ключ.
- Также мы настроили DMARC, который указывает, что делать с письмами, которые провалили проверку SPF и DKIM. RFC 7489 описывает функционирование DMARK.
- SRV-записи нужны для более простого обнаружения сервисов, запущенных клиентами.
- CAA говорит, какие удостоверяющие центры (CA) могут выписывать сертификаты для нашего домена.

Как видите, тут отсутствует PTR-запись и нет возможности ее создать. Проверим, кому принадлежит IP-адрес, выданный нашему серверу:

```
[user@testserver ~]$ dig +short mail.nodrop.in

89.208.87.64

[user@testserver ~]$ whois 89.208.87.64

% This is the RIPE Database query service.

% The objects are in RPSL format.

%

% The RIPE Database is subject to Terms and Conditions.

% See http://www.ripe.net/db/support/db-terms-conditions.pdf


% Note: this output has been filtered.

%       To receive output for a database update, use the "-B" flag.


% Information related to '89.208.84.0 - 89.208.87.255'


% Abuse contact for '89.208.84.0 - 89.208.87.255' is 'abuse@corp.mail.ru'


inetnum:        89.208.84.0 -
89.208.87.255 netname:        RU-
NETBRIDGE-20060418 country:       RU
org:            ORG-LLCn4-RIPE admin-c:
MAIL-RU tech-c:         MAIL-RU status:
ALLOCATED PA mnt-by:         MNT-
NETBRIDGE
```

```
mnt-by:         RIPE-NCC-HM-MNT

created:        2019-03-18T10:38:54Z

last-modified:  2019-03-18T10:38:54Z

source:         RIPE


organisation:    ORG-LLCn4-RIPE  org-name:           Limited

liability  company  Mail.Ru  org-type:           LIR  address:

Leningradskiy  prospect,  39,  build  79,  SkyLight  address:

125167 address:      Moscow address:      RUSSIAN FEDERATION

phone:            +74957256357 fax-no:          +74957256359

abuse-c:        MAIL-RU mnt-ref:        RIPE-NCC-HM-MNT mnt-

ref:        MNT-NETBRIDGE mnt-by:        RIPE-NCC-HM-MNT mnt-

by:         MNT-NETBRIDGE created:       2008-07-24T10:46:43Z

last-modified:  2016-06-23T09:47:53Z source:          RIPE #

Filtered


role:           MAIL.RU NOC address:
Limited liability company Mail.Ru address:
Leningradskiy prospect, 39/79 address:
125167 Moscow Russia phone:          +7 495
7256357 fax-no:         +7 495 7256359
```

```
remarks:        ----------------------------------------
-admin-c:         VG659-RIPE admin-c:         EY1327-RIPE
tech-c:          DBF3-RIPE tech-c:          IS13 remarks:
----------------------------------------remarks:
General questions: noc@corp.mail.ru remarks:        Spam
& Abuse: abuse@corp.mail.ru remarks:        Search Abuse:
search-abuse@corp.mail.ru  remarks:               Routing
inquiries: ncc@corp.mail.ru remarks:      Peering issues:
ncc@corp.mail.ru remarks:        ----------------------
-----------------remarks:        --------- A T T E N T I
O  N  !!!  --------remarks:                  Please  use
abuse@corp.mail.ru e-mail remarks:       address for spam
and abuse complaints. remarks:          Mails for other
addresses will be ignored! remarks:        -------------
---------------------------mnt-by:          MNT-NETBRIDGE
abuse-mailbox:  abuse@corp.mail.ru nic-hdl:        MAIL-
RU created:          2010-11-29T12:03:47Z last-modified:
2018-11-14T20:30:14Z source:         RIPE # Filtered
```

Как видно из вывода, IP-адрес принадлежит Mail.ru Group, следовательно, PTR-запись должна быть прописана на DNS-сервере mail.ru. То есть, необходимо связаться с поддержкой mail.ru и попросить их добавить запись, что и было сделано.

```
[user@testserver ~]$ dig -x 89.208.87.64
```

```
; <<>> DiG 9.11.3-1ubuntu1.5-Ubuntu <<>> -x 89.208.87.64
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 52426
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;64.87.208.89.in-addr.arpa.     IN      PTR

;; ANSWER SECTION:
64.87.208.89.in-addr.arpa. 3600 IN      PTR     mail.nodrop.in.

;; Query time: 71 msec
;; SERVER: 192.168.1.1#53(192.168.1.1)
;; WHEN: Sun Jun 09 21:18:51 DST 2019
;; MSG SIZE  rcvd: 82
```

Установка Mailcow

Мы будем устанавливать Mailcow в папку /home/centos. Для начала убедимся, что umask — 0022, а docker-сервис запущен:

```
[centos@mail ~]$ umask -S 0022 u=rwx,g=rx,o=rx
[centos@mail ~]$ umask
0022
[centos@mail ~]$ sudo systemctl start docker
```

Клонируем Mailcow из Git:

```
[centos@mail ~]$ git clone https://github.com/mailcow/mailcow-dockerized
```

Переходим в соответствующую папку и запускаем генератор конфига:

```
[centos@mail ~]$ git clone https://github.com/mailcow/mailcow-dockerized

[centos@mail ~]$ cd mailcow-dockerized/

[centos@mail mailcow-dockerized]$ ./generate_config.sh
```

Далее отвечаем на вопросы в зависимости от параметров:

```
[centos@mail mailcow-dockerized]$ ./generate_config.sh

Press enter to confirm the detected value '[value]' where applicable or enter a
custom value.

Mail server hostname (FQDN) - this is not your mail domain, but your mail
servers hostname: mail.nodrop.in

Timezone [UTC]: UTC

Installed memory is <= 2.5 GiB. It is recommended to disable ClamAV to prevent
out-of-memory situations.

ClamAV can be re-enabled by setting SKIP_CLAMD=n in mailcow.conf.

Do you want to disable ClamAV now? [Y/n] Y Disabling Solr on low-memory system.
```

Результат ответов на вопросы — файл mailcow.conf, который можно модифицировать, если требуется:

```
[centos@mail mailcow-dockerized]$ ll | grep mail

-rw-------. 1 centos centos  4109 Jun  9 19:32 mailcow.conf
```

Например, мы будем использовать SSL с сертификатами, выписанными бесплатным сервисом letsencrypt. Для этого мы должны в файле mailcow.conf указать доменные имена, на которые следует выписать сертификат. За это отвечает параметр ADDITIONAL_SAN файла mailcow.conf.

```
[centos@mail mailcow-dockerized]$ vim mailcow.conf

ADDITIONAL_SAN=mail.nodrop.in,smtp.nodrop.in,imap.nodrop.in,www.nodrop.in
```

После чего скачиваем все контейнеры:

```
[centos@mail mailcow-dockerized]$ docker-compose pull
```

Скачиваем WebUI для работы с почтой пользователям. В нашем случае это RoundCube. Переходим в папку data/web (полный путь в нашем случае /home/centos/mailcow-dockerized/data/web), скачиваем и распаковываем RoundCube:

```
[centos@mail mailcow-dockerized]$ cd data/web
```

```
[centos@mail web]$ pwd

/home/centos/mailcow-dockerized/data/web

[centos@mail web]$ wget -O - https://github.com/roundcube/roundcubemail/releases/download/1.3.9/roundcubemail -1.3.9-complete.tar.gz | tar xfvz -
```

Переименовываем полученную папку roundcubemail-1.3.9/ в папку rc и меняем права для папки с roundcube, в нашем случае — на CentOS:

```
[centos@mail web]$ mv roundcubemail-1.3.9/ rc

[centos@mail web]$ sudo chown -R centos: rc
```

Создаём файл конфига data/web/rc/config/config.inc.php (полный путь — /home/centos/mailcow-dockerized/data/web/rc/config/config.inc.php) со следующими настройками:

```
[centos@mail web]$ vim /home/centos/mailcow-dockerized/data/web/rc/config/config.inc.php



<?php error_reporting(0); if

(!file_exists('/tmp/mime.types')) {

file_put_contents("/tmp/mime.types",
fopen("http://svn.apache.org/repos/asf/httpd/httpd/trunk/docs/conf/mime.types",
'r'));

}

$config = array();
```

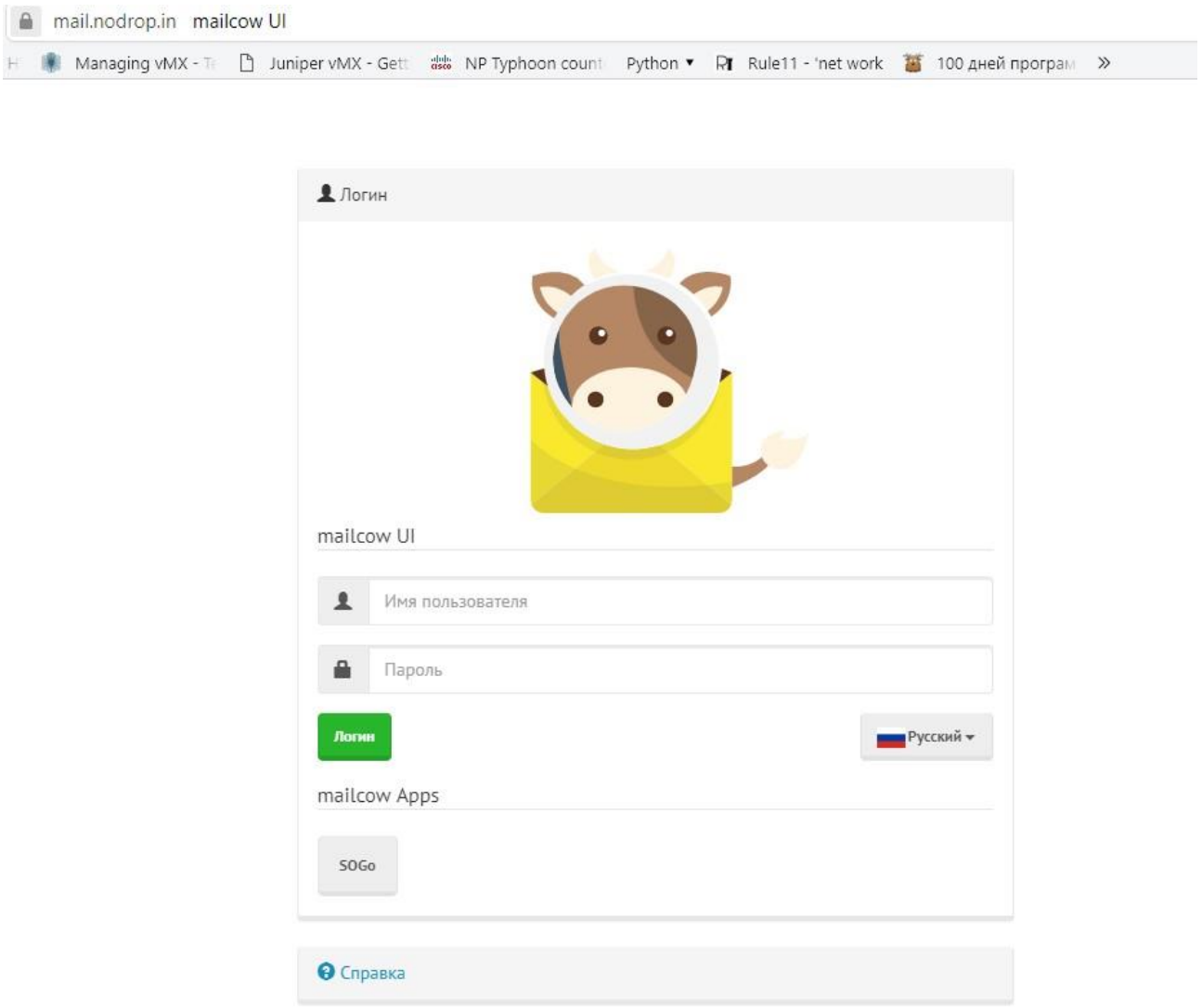
```
$config['db_dsnw'] = 'mysql://' . getenv('DBUSER') . ':' . getenv('DBPASS') .
'@mysql/' . getenv('DBNAME');

$config['default_host'] = 'tls://dovecot';

$config['default_port'] = '143';

$config['smtp_server'] = 'tls://postfix';

$config['smtp_port'] = 587;

$config['smtp_user'] = '%u';

$config['smtp_pass'] = '%p';

$config['support_url'] = '';

$config['product_name'] = 'Roundcube Webmail';

$config['des_key'] = 'yourrandomstring_changeme';

$config['log_dir'] = '/dev/null';

$config['temp_dir'] = '/tmp';

$config['plugins'] = array(

  'archive',

  'managesieve'

);

$config['skin'] = 'larry';

$config['mime_types'] = '/tmp/mime.types';

$config['imap_conn_options'] = array(

            'ssl' => array('verify_peer' => false, 'verify_peer_name' => false,
'allow_self_signed' => true)

);

$config['enable_installer'] = true;

$config['smtp_conn_options'] = array(

            'ssl' => array('verify_peer' => false, 'verify_peer_name' => false,
'allow_self_signed' => true)

);



$config['managesieve_port'] = 4190;


$config['managesieve_host'] = 'tls://dovecot';
```

```
$config['managesieve_conn_options'] = array(

            'ssl' => array('verify_peer' => false, 'verify_peer_name' => false,
'allow_self_signed' => true)

);

// Enables separate management interface for vacation responses (out-of-office)

// 0 - no separate section (default),

// 1 - add Vacation section,

// 2 - add Vacation section, but hide Filters section

$config['managesieve_vacation'] = 1;
```

Запускаем почтовый сервер:

```
[centos@mail web]$ cd /home/centos/mailcow-dockerized/

[centos@mail mailcow-dockerized]$ docker-compose up -d
```

Как только сервис стартовал, у нас появляется возможность попасть в админскую часть нашего почтового сервера, используя логин admin и пароль moohoo:



После чего завершаем настройку roundcube, перейдя по адресу https://your_servername/rc/installer:

```
https://nodrop.in/rc/installer
```

roundcube
open source webmail software

Roundcube Webmail Installer

1. **Check environment** 2. Create config 3. Test config

Checking PHP version

Version: **OK** (PHP 7.3.5 detected)

Checking PHP extensions

The following modules/extensions are *required* to run Roundcube:

PCRE: **OK**
DOM: **OK**
Session: **OK**
XML: **OK**
JSON: **OK**
PDO: **OK**
Multibyte: **OK**
OpenSSL: **OK**

The next couple of extensions are *optional* and recommended to get the best performance:

FileInfo: **OK**
Libiconv: **OK**
Intl: **OK**
Exif: **NOT AVAILABLE** (See http://www.php.net/manual/en/book.exif.php)
LDAP: **OK**
GD: **OK**
Imagick: **OK**

Checking available databases

Check which of the supported extensions are installed. At least one of them is required.

MySQL: **OK**
PostgreSQL: **NOT AVAILABLE** (See http://www.php.net/manual/en/ref.pdo-pgsql.php)
SQLite: **OK**
SQLite (v2): **NOT AVAILABLE** (See http://www.php.net/manual/en/ref.pdo-sqlite.php)
SQL Server (SQLSRV): **NOT AVAILABLE** (See http://www.php.net/manual/en/ref.pdo-sqlsrv.php)
SQL Server (DBLIB): **NOT AVAILABLE** (See http://www.php.net/manual/en/ref.pdo-dblib.php)
Oracle: **NOT AVAILABLE** (See http://www.php.net/manual/en/book.oci8.php)

Check for required 3rd party libs

This also checks if the include path is set correctly.

PEAR: **OK**
Auth_SASL: **OK**
Net_SMTP: **OK**
Net_IDNA2: **OK**
Mail_mime: **OK**
Net_LDAP3: **OK**

Checking php.ini/.htaccess settings

The following settings are *required* to run Roundcube:

file_uploads: **OK**
session.auto_start: **OK**
mbstring.func_overload: **OK**
suhosin.session.encrypt: **OK**

The following settings are *optional* and recommended:

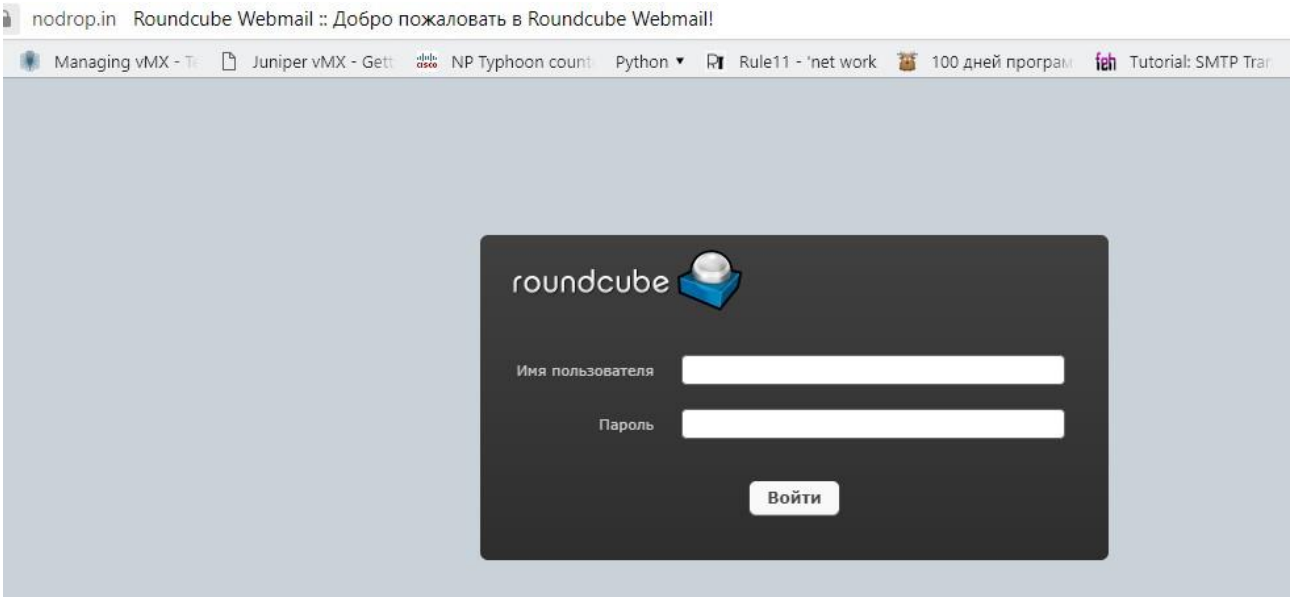
allow_url_fopen: **OK**
date.timezone: **OK**

NEXT

Далее нажимаем next-> next и завершаем настройку Roundcube. И не забываем в конце удалить папку data/web/rc/installer после успешной установки roundcube.

GUI будет доступен по адресу:

```
https://nodrop.in/rc/
```



Практическое задание

В качестве практического задания вам предлагается дома, используя уже существующий DNS-сервер, настроить контейнеризированую почту maicow. Используйте полную инструкцию по настройке mailcow.

Список литературы

Для подготовки данного методического пособия были использованы следующие ресурсы:

1. Mailcow.email.
2. Настройка DKIM/SPF/DMARC-записей, или Защищаемся от спуфинга.
3. Linux Email: Set up and Run a Small Office Email Server. Patrick Ben, Magnus Back, Alistair McDonald, Carl Taylor, David Rusenko. ISBN: 9781904811374