Replicating Replication

highly available, open source Dovecot 2.4 servers

Jens Erat

January 31st, 2026

Outline

- Why
- Replication in Dovecot 2.3
- High Availability Options in Dovecot 2.4
- Dovewarden

Why

Commit 4c04e4c

```
commit 4c04e4c30fd4817a8b0e11d04d9681173f696f41
Author: Aki Tuomi <aki.tuomi@open-xchange.com>
Date:   Thu Jun 29 18:11:26 2023 +0300

    global: Remove replicator

diff --git a/configure.ac b/configure.ac
index f09d0254ad..74090b44a1 100644
--- a/configure.ac
+++ b/configure.ac
@@ -804,9 +804,6 @@ src/pop3/Makefile
 src/pop3-login/Makefile
 src/submission/Makefile
 src/submission-login/Makefile
-src/replication/Makefile
-src/replication/aggregator/Makefile
-src/replication/replicator/Makefile
[...]
```

Me

- 2024-now
  Kubernetes fleet management at Deutsche Telekom
- 2017-2023
  Kubernetes fleet management at Mercedes-Benz

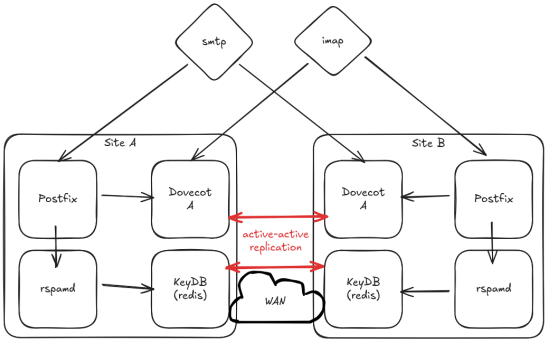
- until 2016
  Master in Computer Sciences at University of Constance

Me

- 2024-now
  Kubernetes fleet management at Deutsche Telekom
- 2017-2023
  Kubernetes fleet management at Mercedes-Benz
- 2014-2016
  Groupware and Mail team at University of Constance
- until 2016
  Master in Computer Sciences at University of Constance

My mail setup

- running my own mail server
- obviously, hosted on of Kubernetes
  - multi-site setup behind DSL lines (high latency)
- Dovecot replication as perfect match for active-active high availability



Replication in Dovecot 2.3

dsync

`doveadm sync -u username@example.com remote:server-replica.example.com` [1]

- mailbox synchronization between Dovecot and other IMAP server
- uni- and bidirectional sync
  - conflict resolution: allows changes to mailbox on both sides
- very useful for any kind of migration tasks
- stateful incremental sync very efficient, also for large mailboxes

- was used as basis for Dovecot replication
- layer 7 replication, no shared storage needed
  - eliminates single point of failure
  - works well over high latency links
- still available in Dovecot 2.4 with full feature set

- we just need to call sync "whenever needed" to rebuild replication?

[1] https://doc.dovecot.org/2.4.2/core/man/doveadm-sync.1.html

dsync

`doveadm sync -u username@example.com remote:server-replica.example.com` [1]

- mailbox synchronization between Dovecot and other IMAP server
- uni- and bidirectional sync
  - conflict resolution: allows changes to mailbox on both sides
- very useful for any kind of migration tasks
- stateful incremental sync very efficient, also for large mailboxes

- was used as basis for Dovecot replication
- layer 7 replication, no shared storage needed
  - eliminates single point of failure
  - works well over high latency links
- still available in Dovecot 2.4 with full feature set

- we just need to call sync "whenever needed" to rebuild replication?

[1] https://doc.dovecot.org/2.4.2/core/man/doveadm-sync.1.html

dsync

`doveadm sync -u username@example.com remote:server-replica.example.com` [1]

- mailbox synchronization between Dovecot and other IMAP server
- uni- and bidirectional sync
  - conflict resolution: allows changes to mailbox on both sides
- very useful for any kind of migration tasks
- stateful incremental sync very efficient, also for large mailboxes

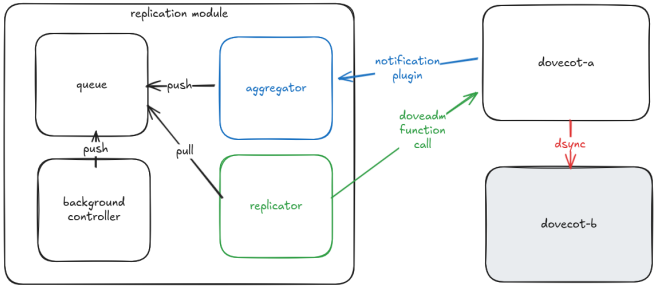
- was used as basis for Dovecot replication
- layer 7 replication, no shared storage needed
  - eliminates single point of failure
  - works well over high latency links
- still available in Dovecot 2.4 with full feature set

- we just need to call sync “whenever needed” to rebuild replication?

[1] https://doc.dovecot.org/2.4.2/core/man/doveadm-sync.1.html

Replication Module

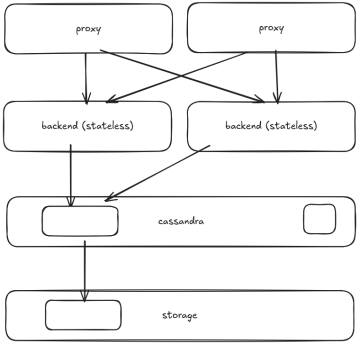
- **notification plugin**: sends events for mailbox changes (and others)
- **aggregator**: collect events and add users to queue
- **replicator**: process queue and call dsync



High Availability Options in Dovecot 2.4

Dovecot Pro

- commercial offering
- also doesn't have replication daemon
- makes backend stateless
- obox as new storage backend
  - metadata in cassandra
  - mail data in storage system (eg. object storage)



Shared Filesystem

> ***Dovecot allows keeping mails*** *and index files in* ***clustered filesystems****. Dovecot does not specifically support any specific clustered solution - it is the responsibility of the admin to perform functional and load testing to guarantee the storage solution provides adequate performance.*

- NFS as de facto standard
- special challenges with other clustered filesystems
- single point of failure, latency critical on network path (bound to metro-level geo redundancy)

Result: can be acceptable in some scenarios, but not ideal

Carrying over the patch

Patrick Cernko provided a patch to reintroduce replication daemon in Dovecot 2.4.
Project page
https://codeberg.org/errror/wormhole

Advantages:

- uses existing, well tested dsync and replication code
- compatible with existing configuration
- drop-in replacement

Disadvantages:

- requires patching Dovecot source code
- cannot use official Dovecot packages
- no upstream support
- effort carrying over the patch will increase over time

Replication Daemon

dsync is still around.

we just need to call sync "whenever needed" to rebuild replication?

When is "whenever needed"?

- cronjob either too late or huge overhead

notification system

- Dovecot event API to the rescue, receive notifications via HTTP webhook

queue

- preferably priority queue: new mails more important than marking a mail as read
- Redis as simple, reliable queue backend

replication controller

- receive notifications, enqueue jobs
- dequeue jobs, call dsync
- error handling
- background replication
- some hundred lines of Go code

Replication Daemon

dsync is still around.

we just need to call sync "whenever needed" to rebuild replication?

When is "whenever needed"?

- cronjob either too late or huge overhead

notification system

- Dovecot event API to the rescue, receive notifications via HTTP webhook

queue

- preferably priority queue: new mails more important than marking a mail as read
- Redis as simple, reliable queue backend

replication controller

- receive notifications, enqueue jobs
- dequeue jobs, call dsync
- error handling
- background replication
- some hundred lines of Go code

Replication Daemon

dsync is still around.

> *we just need to call sync "whenever needed" to rebuild replication?*

When is "whenever needed"?

- cronjob either too late or huge overhead

notification system

- Dovecot event API to the rescue, receive notifications via HTTP webhook

queue

- preferably priority queue: new mails more important than marking a mail as read
- Redis as simple, reliable queue backend

replication controller

- receive notifications, enqueue jobs
- dequeue jobs, call dsync
- error handling
- background replication
- some hundred lines of Go code

Replication Daemon

dsync is still around.

> *we just need to call sync "whenever needed" to rebuild replication?*

When is "whenever needed"?

- cronjob either too late or huge overhead

notification system

- Dovecot event API to the rescue, receive notifications via HTTP webhook

queue

- preferably priority queue: new mails more important than marking a mail as read
- Redis as simple, reliable queue backend

replication controller

- receive notifications, enqueue jobs
- dequeue jobs, call dsync
- error handling
- background replication
- some hundred lines of Go code

Dovewarden

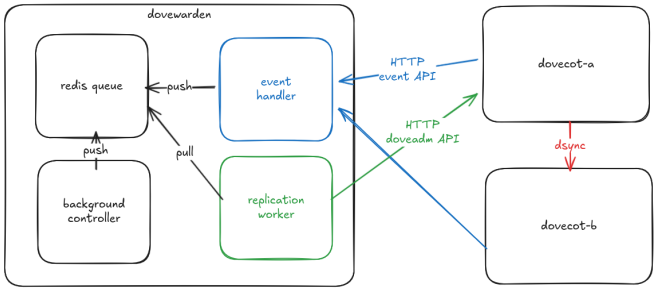
Dovewarden

Architecture

- dovewarden as replication daemon
- communication over HTTP APIs

- event configuration in Dovecot
  - `imap_command_finished` and command filter
  - `mail_delivery_finished`
- unmodified Dovecot release

- built-in miniredis for small setups
- external redis allows scale-out

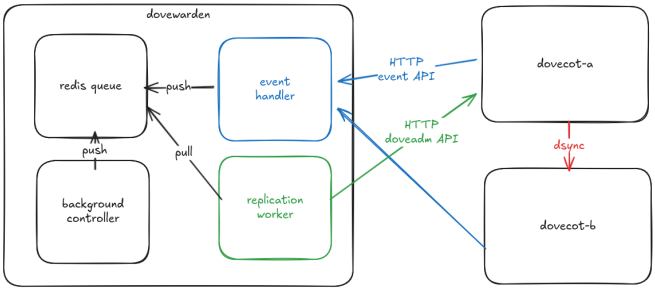


Architecture

- dovewarden as replication daemon
- communication over HTTP APIs

- event configuration in Dovecot
  - `imap_command_finished` and command filter
  - `mail_delivery_finished`
- unmodified Dovecot release

- built-in miniredis for small setups
- external redis allows scale-out

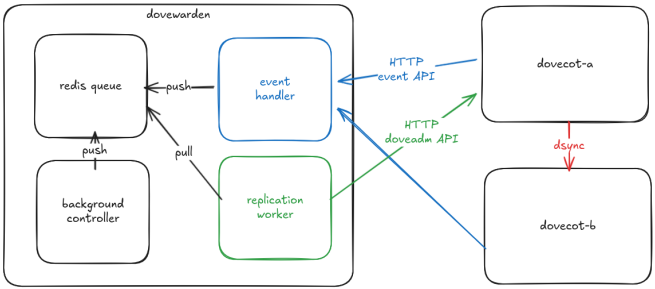


Architecture

- dovewarden as replication daemon
- communication over HTTP APIs

- event configuration in Dovecot
  - `imap_command_finished` and command filter
  - `mail_delivery_finished`
- unmodified Dovecot release

- built-in miniredis for small setups
- external redis allows scale-out



Features

External notification daemon

- will not crash your Dovecot server
- standard API usage, low risk of major changes in Dovecot

Notification receiver

- full coverage of IMAP commands
- mail delivery

Extensive test suite

- automated E2E test suite using real Dovecot servers
- high unit test coverage

Queue

- batteries included: built-in miniredis for testing and small setups
- but swappable: optional external Redis backend

Observability

- OpenMetrics (Prometheus) data collection endpoint
- error rates, queue lengths, replication durations, . . .

Docker image and helm chart

- easy deployment on Kubernetes clusters

Features

External notification daemon

- will not crash your Dovecot server
- standard API usage, low risk of major changes in Dovecot

Notification receiver

- full coverage of IMAP commands
- mail delivery

Extensive test suite

- automated E2E test suite using real Dovecot servers
- high unit test coverage

Queue

- batteries included: built-in miniredis for testing and small setups
- but swappable: optional external Redis backend

Observability

- OpenMetrics (Prometheus) data collection endpoint
- error rates, queue lengths, replication durations, . . .

Docker image and helm chart

- easy deployment on Kubernetes clusters

Features

External notification daemon

- will not crash your Dovecot server
- standard API usage, low risk of major changes in Dovecot

Notification receiver

- full coverage of IMAP commands
- mail delivery

Extensive test suite

- automated E2E test suite using real Dovecot servers
- high unit test coverage

Queue

- batteries included: built-in miniredis for testing and small setups
- but swappable: optional external Redis backend

Observability

- OpenMetrics (Prometheus) data collection endpoint
- error rates, queue lengths, replication durations, . . .

Docker image and helm chart

- easy deployment on Kubernetes clusters

Features

External notification daemon

- will not crash your Dovecot server
- standard API usage, low risk of major changes in Dovecot

Notification receiver

- full coverage of IMAP commands
- mail delivery

Extensive test suite

- automated E2E test suite using real Dovecot servers
- high unit test coverage

Queue

- batteries included: built-in miniredis for testing and small setups
- but swappable: optional external Redis backend

Observability

- OpenMetrics (Prometheus) data collection endpoint
- error rates, queue lengths, replication durations, . . .

Docker image and helm chart

- easy deployment on Kubernetes clusters

Features

External notification daemon

- will not crash your Dovecot server
- standard API usage, low risk of major changes in Dovecot

Notification receiver

- full coverage of IMAP commands
- mail delivery

Extensive test suite

- automated E2E test suite using real Dovecot servers
- high unit test coverage

Queue

- batteries included: built-in miniredis for testing and small setups
- but swappable: optional external Redis backend

Observability

- OpenMetrics (Prometheus) data collection endpoint
- error rates, queue lengths, replication durations, ...

Docker image and helm chart

- easy deployment on Kubernetes clusters

Features

External notification daemon

- will not crash your Dovecot server
- standard API usage, low risk of major changes in Dovecot

Notification receiver

- full coverage of IMAP commands
- mail delivery

Extensive test suite

- automated E2E test suite using real Dovecot servers
- high unit test coverage

Queue

- batteries included: built-in miniredis for testing and small setups
- but swappable: optional external Redis backend

Observability

- OpenMetrics (Prometheus) data collection endpoint
- error rates, queue lengths, replication durations, ...

Docker image and helm chart

- easy deployment on Kubernetes clusters

Dovecot configuration

```
event_exporter dovewarden {
  driver = http-post
  http_post_url = http://dovewarden:8080/events

  format = json
  time_format = rfc3339
}

metric dovewarden {
  exporter = dovewarden
  filter = event=mail_delivery_finished OR ( \
  event=imap_command_finished AND tagged_reply_state=OK AND category=service:imap AND ( \
    cmd_name="APPEND" or \
    cmd_name="COPY" or \
    cmd_name="CLOSE" or \
    cmd_name="CREATE" or \
    [...]
  ))
}
```

Available now

Homepage: **dovewarden.org**
Code: **github.com/dovewarden/dovewarden**

- available now
- running in production at my mail server
- tested compatible with Dovecot 2.4

Call for action

Missing for v1

- assumed compatible with Dovecot 2.3, but not yet tested
- load testing
- real-world testing on larger systems
- some more metrics, example dashboard+alerts
- better documentation

What's your needs?

- non-containerized deployment options
- is POP3 still around?
- what's your preferred replication transport?
- **your feedback!**

Contact

https://www.dovewarden.org
email@jenserat.de – xmpp: jabber@jenserat.de