RYOMS Excerpt for Apple Debugging

Michael W Lucas

# Copyright Information

Run Your Own Mail Server

Copyright 2024 by Michael W Lucas (https://mwl.io).

Author: Michael W Lucas

Copyeditor: Amanda Robinson

Cover art: Eddie Sharam

ISBN (paperback): 978-1-64235-078-4

ISBN (hardcover): 978-1-64235-079-1

All rights reserved. No part of this work may be reproduced or transmitted in any form or by any means, electronic or mechanical, including but not limited to photocopying, recording, or by any information storage or retrieval system, without the prior written permission of the copyright holder and the publisher. For information on book distribution, translations, or other rights, please contact Tilted Windmill Press (accounts@tiltedwindmillpress.com).

The information in this book is provided on an “As Is” basis, without warranty. While every precaution has been taken in the preparation of this work, neither the author nor Tilted Windmill Press shall have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the information contained in it.

Tilted Windmill Press

https://www.tiltedwindmillpress.com

# Chapter 0: Introduction

Email is the heart of the Internet. Online forums and social media and chat systems have blossomed and thrived with the web, but like dedicated Star Wars fans we keep returning to the classics. Email works everywhere with an Internet connection, and in many places without. Unlike chats, where the other people involved see that you’ve received their message and are choosing to ignore them, you answer email when you feel like it. Your correspondents don’t get a notification that you’re typing a reply and watch the screen until you compose words. Email is one of the few surviving asynchronous communication tools available on every platform.

The first email-like message was sent in 1965 at MIT, when the Internet was only a drunken dream of comp sci grad students. When those students finally connected two mainframes, they developed the primordial Internet protocols so the machines could communicate and a program called MAILBOX. Today’s email looks nothing like theirs—for one, email addresses have those esoteric @ symbols in them rather than nice simple exclamation points. Modern email is built on the lessons learned from those early protocols, however. That’s not bad—all broadly used protocols evolved the same way. Today, a handful of providers dominate email services. Companies like Google and Microsoft control most email addresses. They’re an Email Empire, while those of us who run our own email systems are a ragtag gang struggling for independence.

It can be done. The protocols are comprehensible, the software freely available, the debugging tools adequate, and the hardware affordable.

The problem with running email has nothing to do with the technology. Once you know how, you can set up an email server by hand in minutes—or in seconds with automation. The problem is that the email server is only one part of running an email server.

Consider setting up your first web site from scratch. You install the software, start the daemon, and point your web browser at the host’s IP address. You get the default page and a warm thrill of victory. You have a web server! If you put content on it, you can tell your colleagues “Hey, browse to 203.0.113.99 and you’ll find my new web site!” You have made a Thing. Add an entry in the Domain Name Service (DNS) and you can offer folks a link like http://mwl.io. More testing and you add TLS, granting you the precious S in https. Step by step, you build a web site that complies with modern standards.[[1]](#footnote-1)

Email isn’t like that.

Running email in the real world is not a configuration problem. It is about citizenship. Once the Email Empire declares your server inadequate or untrustworthy, it’s very hard to get them to change their mind. A spammer’s mark is not an easy thing to live with. You must prepare to solidly participate in email society before sending your first message to the outside world. Sure, you can set up a couple of throwaway hosts on expendable domains and fling email exclusively between them to see how the software works, but if you want to run a real email server and you want other organizations to accept your email, you must quickly establish a good reputation. You can’t get basic SMTP up and immediately start sending messages to accounts in the Email Empire; they’ll declare you untrustworthy. It’s only safe to send mail to those accounts once you have fully configured SPF, DKIM, and DMARC. While the Empire often claims it needs less than that, they also reject messages because “they feel like it.”

Once you establish your system as a good citizen that follows standard practices, operating a mail server is comparatively low maintenance. Most of the time I spend on my email is spent applying security patches, same as any other Internet-facing application. Running a mail server for a large corporation is a full-time job, yes, but most of that is spent explaining to users that an error message of “The recipient’s email account is over quota” means that the email bounced because the recipient hasn’t cleaned out their inbox.

Run Your Own Mail Server uses common freely available software to illustrate the mechanics of email, but the book’s main focus is on establishing that citizenship. You can use any software you like that fulfills the necessary roles. This book does not cover every possible use case across every platform. It focuses on what everyone must do to support a mail system that can function on the modern Internet. You will learn how the components work together, and be able to research and add less common features that you need.

## Who Should Read This Book

Read this book if you want to run an email server for a small organization, for yourself and your friends, or just you. This book will not prepare you to run email for a global enterprise or a large mailing list, but it will provide a foundation for you to study those advanced cases.

Most people with the background necessary for running their own mail system have several domain names, one or two organizations that they’d like to help, and perhaps even a friend. This book is designed to let you support those folks in a scalable manner. The common sysadmin tendency to offer too much help, become overloaded, and implode is left for you to cope with.

## Ethics of System Administration

I have firm memories of the first time I got root on a mail server, back when the 1.544Mb T1 was the pinnacle of connectivity and my 33.6K modem thrilled me. I thought, “I can’t possibly view everybody’s mail.” I ran more /var/mail/boss and immediately recoiled, guts churning in horror of my appalling power.

Whoever controls a server can see all the information stored on that server. Every system administrator knows this, but in most cases extracting the information is too much trouble to bother. Maybe you’re the database administrator for your organization’s ecommerce system, but you’re probably far more interested in how much space the database uses than who bought what.

Until you’re not.

Unlike a database, examining email stored as text files is trivial. You don’t even need SQL, only privilege and a pager.

When you run a service, the people who use that service are trusting you. System administrators have no formal professional code of ethics, but organizations and society have privacy standards. I am not a professional ethicist, nor do I have a degree in philosophy, but society’s overwhelming consensus is that snooping is wrong. Most organizations consider privacy violations to be against their policies and will terminate you for it. Your spouse might terminate your relationship for it. Just as the folks in HR shouldn’t dig into your personnel file to satisfy their puerile curiosity, you shouldn’t dig into someone’s email except in response to an actual issue.

Sometimes you must examine a user’s email. A long time ago, at an employer far far away, I had to troubleshoot a corrupt mailbox. A sender had innocently attached a word processor document to a message, and something in that document confused the POP3 server. I had to examine the individual characters in the mailbox file to identify the problem, which meant reading the user’s email. I asked the user for permission to examine their mail. They granted it. I saw parts of a legal document I had no business seeing, identified the issue, extracted the problem document from the mailbox, and restored service.[[2]](#footnote-2)

Why is that different? Consent. If the user had refused permission I would have had to escalate to our boss and let him sort through the personnel and privacy issues.

While most of email’s rough edges have been burnished off since, violating privacy during troubleshooting remains a real possibility. Test users, full-on testing environments, and program analyzers like dtrace(1) are inexpensive. Use them.

## Why Bother?

If email is such a pain, why run it yourself?

Privacy, control, and education.

Service providers often scan email for advertising keywords or saleable personal data. While ad blockers keep the ads out of your eyeballs, they solve the problem at the wrong end. Yes, if you email someone with an Email Empire address the Empire adds your content to their records, but when you correspond with other outsiders they learn nothing. Email can be maliciously intercepted, but that interception takes work and, for the most part, service providers don’t bother.

Privacy is especially important in business. A service provider might promise not to mine your company emails, but one of their sysadmins trying to fix your issues might have to examine your email. If your organization’s privacy needs violating, it’s better someone in your organization does it.

You control your own email. Many service providers offer fancy dashboards that let you add and delete users or set quotas, but when something goes sideways you’re stuck working with their support department. Waiting for a tech to claim your trouble ticket is infuriating, especially when you want them to do something trivial like check the log. When you run your own email, you control the problem.

Perhaps the least valid but most important reason for many is pure “geek cred.” We want to know how things work. Just as building your own firewall is a great way to learn about networking, running your own mail server is the best way to understand this ecosystem. You run your own web server, DNS, cloud storage and home automation, why not run your own email?

## Prerequisites

This is a book for beginning email operators, but it’s not a book for a beginner sysadmin. You must have certain skills before you can hope to run an email server. Experienced mail administrators can probably work around many of the requirements below, but beginners shouldn’t make their education any more difficult than necessary.

You must have used email. I will not explain what email addresses and attachments are.

You must know how to manage your operating system. You need to back up your hosts, apply security updates, know the difference between network and local sockets, and in general make your hosts fit to sit on the naked Internet. This book assumes and recommends open-source software. I demonstrate everything with Postfix running on FreeBSD or Debian Linux. I list all my configuration files and directories under /etc. That might not reflect your operating system’s defaults. If I say */etc/redis.conf*, you need to figure out that your operating system uses */etc/redis/redis.conf* or */usr/local/etc/redis.conf* or whatever. I provide the context for all my configuration choices so you can make your own decisions on any Unix-like platform or even your preferred mail server.

A mail system is not a single application. It is a system of interlocking applications. Every time you change anything, you must verify that the new part works and that you didn’t break the rest of the system. If you’re not comfortable double-checking your work and testing each stage of a deployment, become comfortable.

You need at least two test hosts attached to the Internet. Yes, hosts can often send mail to accounts on the local host, but that’s worked fine for decades without trouble. These could be virtual machines out in the cloud, hosts in your lab, or whatever. Tests inflict negligible hardware load. These hosts need different static IP addresses. You must also control the world’s access to those addresses, as email requires opening several TCP/IP ports to and from the hosts. Most home Internet providers specifically block these ports.

Those hosts should be in different domains. Yes, you could send email between newyork.solveamurder.org and detroit.solveamurder.org, but your brain absorbs better patterns if you test between wholly unrelated domains like solveamurder.org and ratoperatedvehicle.com. Domains are inexpensive these days. Get two—or set up private DNS and test between example.org and private.test.[[3]](#footnote-3)

You must understand basic DNS, and control the forward and reverse DNS for your test hosts. Supporting email for a domain requires creating and maintaining several DNS entries in that domain. Similarly, the reverse DNS must match the hostname and the forward DNS. If your host advertises its name as mail.solveamurder.org, but reverse DNS identifies your host as customer87.chicago.bighosting.com, your email will go nowhere. When a client looks up the hostname for your IP address, they’ll then check that the forward DNS points to that IP. Any extra hostnames tied to that address are ignored. Yes, a hostname can resolve to several IP addresses, but those IPs must all have reverse DNS that matches the mail server’s hostname.

Be leery of hosting providers that offer limited web interfaces for DNS management. Many of these interfaces limit what sorts of records you can create, or do not permit enough characters for SPF or DKIM records. Before attempting to deploy email, verify that nobody else has trouble entering DKIM or SPF records into your provider’s systems.

1. Hopefully you stop before you install WordPress and join the dark side. Unlike me. [↑](#footnote-ref-1)
2. And, as a result of seeing that document, sent my resume to everyone I knew before the buyout was announced. [↑](#footnote-ref-2)
3. It is the third decade of the twenty-first century, and petrified pettiness means we still have no standards defining specific domains for internal-only use. [↑](#footnote-ref-3)