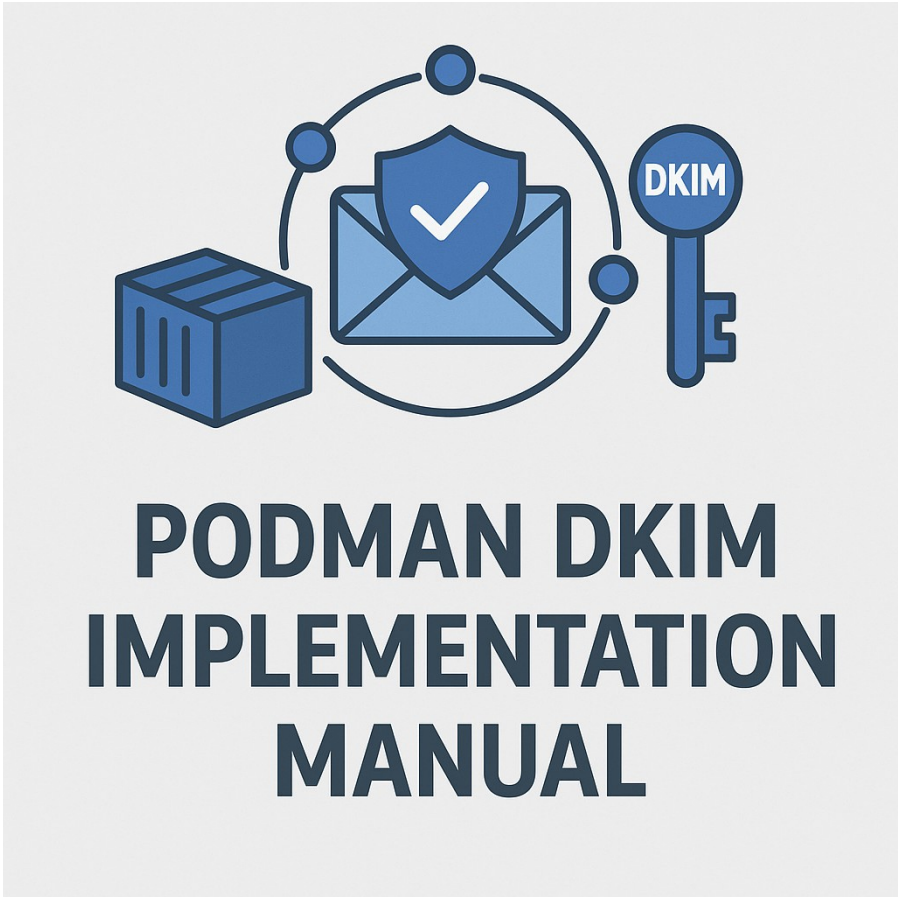


Implementing DKIM Email Signing for GroupWise (mydomain.com)

Table of Contents

SECTION 1 – INTRODUCTION AND OVERVIEW

Title

Implementing DKIM Email Signing for GroupWise (mydomain)
Revision 1.0 – 15 October 2025
Language: Australian English
Environment Host: xxxx`-dkimhost-001 (xxx.xxx.xxx.xxx)`
Domains:

Domain	Selector	Container Name	IP Address
mydomain.com	`myd2025`	dkim-mydomain-com	xxx.xxx.xxx.xxx
example.org	sample2025	dkim-example-org	xxx.xxx.xxx.X

Purpose

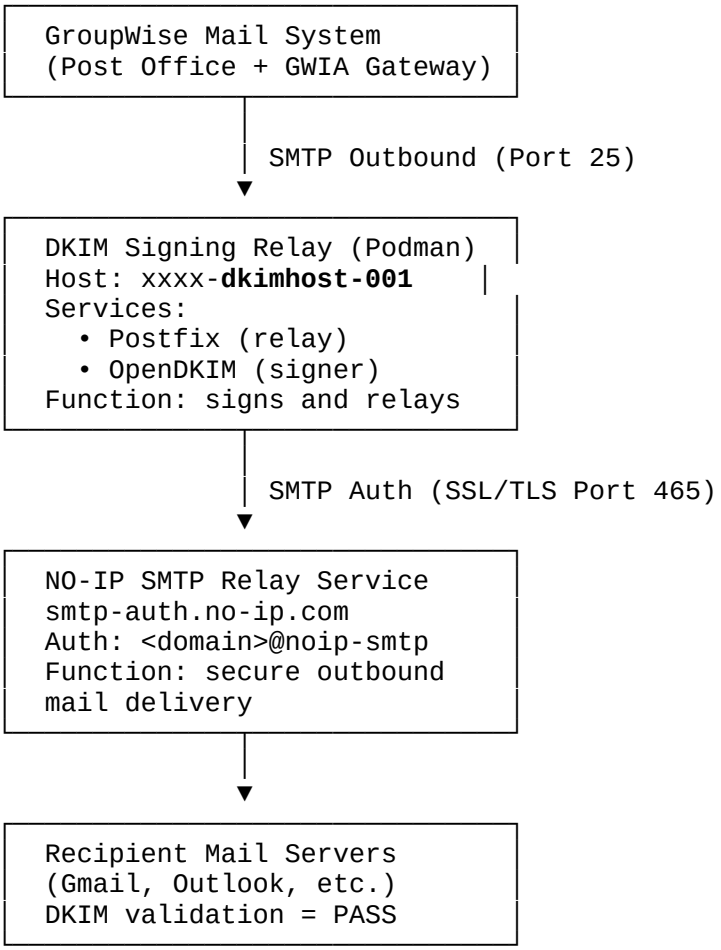
This document provides complete technical and procedural guidance for deploying and maintaining a **DomainKeys Identified Mail (DKIM)** signing relay for GroupWise outbound email.
It enables authenticated, cryptographically signed outbound messages using domain-specific DKIM keys while relaying mail securely through the **NO-IP SMTP service** using SSL/TLS on port 465 (or a standard MTA)

The environment uses **Podman** containers to encapsulate **Postfix** and **OpenDKIM**, providing isolation, reproducibility, and ease of backup.
Each domain is independently signed and routed using its own configuration, stored under `/srv/dkim/<domain>`.

System Overview

GroupWise, via its **GWIA (GroupWise Internet Agent)**, normally sends outbound mail directly to the Internet or an upstream relay.
This configuration introduces a **dedicated DKIM relay layer** that signs each message before relaying through the NO-IP service or directly to the internet.

Mail Flow Architecture (Text Diagram)



Why Use Podman?

Using **Podman** instead of installing Postfix/OpenDKIM directly on the host provides:

- Process isolation and easier rollback (each domain's mail relay is its own container)
- Simpler backup/restore (persistent config under `/srv/dkim/<domain>`)
- Minimal dependency pollution on the base OS
- Easier future migration to Leap 16 or SLES

Components Summary

Component	Purpose
Postfix	Handles SMTP relay and authentication to NO-IP
OpenDKIM	Signs outbound email headers with DKIM keys
Podman	Container runtime managing isolated environments

Component	Purpose
macvlan Network (dkimnet)	Provides each container its own IP address
NO-IP SMTP Relay	Secure outbound SMTP with DKIM-signed traffic

---

Document Structure

This guide is divided into these main sections:

1. System and Network Requirements
2. Environment Preparation
3. Building the Container Environment
4. Domain Configuration
5. Generating DKIM Keys
6. Running Containers
7. Testing and Verification
8. Maintenance and Rotation
9. Appendices (A–H)

SECTION 2 – SYSTEM AND NETWORK REQUIREMENTS

This section defines the hardware, operating system, and network environment required to deploy the DKIM signing relay.
The specifications are designed for long-term reliability, easy backup, and minimal maintenance.

2.1 Hardware Specification

Component	Recommended	Notes
CPU	2 vCPUs (x86_64)	More if handling >10k messages/day
RAM	4 GB	2 GB minimum; 4 GB recommended
Disk	40 GB total	/srv (20 GB), /var (10 GB), / (10 GB)
Network	1 bridged adapter	Must support macvlan subnets
Virtualisation	Any (KVM, XEN, VMware, etc.)	Ensure nested networking supported

💡 *If you host multiple DKIM relays on the same hypervisor, allocate static IPs within a dedicated VLAN for clarity and traceability.*

2.2 Operating System

Component	Value
Distribution	openSUSE Leap 15.6
Kernel	6.x (default)
SELinux	Disabled (AppArmor used)
Firewall	firewalld or nftables – ensure outbound 465 allowed
Time Sync	chronyd or systemd-timesyncd active
Locale	en_AU.UTF-8

Run the following immediately after installation:

```
sudo zypper refresh
sudo zypper update -y
sudo reboot
```

2.3 Host Identity and DNS

Each DKIM host requires a **static hostname** and **A record** in DNS.

Host	IP Address	Purpose
xxxx-dkimhost-001	xxx.xxx.xxx.157	DKIM host for C4Israel

Confirm hostname:

```
hostnamectl set-hostname xxxx-dkimhost-001
```

Verify forward and reverse DNS resolution:

```
getent hosts xxxx-dkimhost-001
```

```
dig -x xxx.xxx.xxx.xxx +short
```

2.4 Network Layout

A dedicated **macvlan network** will be created within Podman to give each DKIM container its own static IP address.

Network Name	Type	Subnet	Gateway	Parent Interface
dkimnet	macvlan	xxx.xxx.xxx.0/24	xxx.xxx.xxx.254	eth0

Each container will attach to this network with a fixed IP:

Container	Domain	IP Address
dkim-mydomain-com	mydomain.com	xxx.xxx.xxx.xxx
dkim-example-org	example.org	xxx.xxx.xxx.xxx

2.5 Firewall and Port Access

Outbound access is required to:

Protocol	Port	Destination	Purpose
TCP	465	smtp-auth.no-ip.com	Secure SMTP relay (SSL/TLS)
TCP	53	DNS servers	DKIM record lookup and relay resolution
TCP	22	Internal admin access only	For SSH to host
TCP	80/443	Optional	OS updates and remote monitoring tools

Inbound access is generally **not required**, as all mail is outbound from GroupWise through GWIA → DKIM Relay → NO-IP.

2.6 DNS Requirements

Each domain must publish:

1. An **A record** pointing to the host (for identification, not mail delivery).
2. A **TXT record** for the DKIM key (added later during key generation).

Example for `mydomain.com`:

Type	Name	Value
TXT	xxx2025._domainkey.mydomain.com	`"v=DKIM1; k=rsa; p=<public-key-string>"`

2.7 Backup and Snapshot Planning

Before proceeding, create host-level snapshots or backups of:

- `/var/Software`
- `/srv`
- `/etc/containers`

These contain all configuration, build files, and container data.

Example using tar:

```
sudo tar czf /backup/pre-dkim-setup-$(date +%F).tar.gz /var/Software /srv /etc/containers
```

2.8 Mail Flow Integration with GroupWise

Adjust GroupWise Internet Agent (GWIA) to send outbound mail through this new relay host:

1. Log into the GroupWise Admin Console.
2. Open **Internet Agent > Access Control**.
3. Set **Relay Host for Outbound Messages** to:

   `dkim-mydomain-com.cdn.mydomain.com:25`
4. Save and restart GWIA.

This routes all external mail through the DKIM host for signing and relay via NO-IP.

SECTION 3 – ENVIRONMENT PREPARATION

This section covers the setup of the host operating system, installation of dependencies, and creation of the Podman DKIM build environment.
It assumes a clean installation of **openSUSE Leap 15.6** and a working network connection.

3.1 Verify Base System

After installation, confirm the system is properly updated and time-synchronised:

```
sudo zypper refresh
sudo zypper update -y
sudo systemctl enable --now chronyd
timedatectl
```

Expected output should show:

```
NTP service: active
Time zone: Australia/Brisbane
```

If you're using virtualisation with snapshots, take a baseline snapshot at this point.
This will provide a rollback point if something goes wrong later.

3.2 Install Core Tools

Install useful utilities for system management and diagnostics:

```
sudo zypper install -y vim htop iotop ncdu net-tools bind-utils curl wget git
```

These are not strictly required for DKIM operation, but they simplify administration.

3.3 Verify Network Connectivity

Check connectivity to internal and external services:

```
ping -c3 xxx.xxx.xxx.254
ping -c3 smtp-auth.no-ip.com
```

Verify DNS resolution:

```
dig smtp-auth.no-ip.com +short
```

Confirm outbound SSL/TLS connectivity to the relay service:

```
openssl s_client -connect smtp-auth.no-ip.com:465 -crlf -quiet
```

Expected result: `CONNECTED(00000003)` and valid certificate output.

3.4 Create Build Directory

All DKIM build assets are stored under `/var/Software/podman_dkim_bundle_leap15_6`.

```
sudo mkdir -p /var/Software
cd /var/Software
sudo tar xzf /path/to/podman_dkim_bundle_leap15_6.tar.gz
cd podman_dkim_bundle_leap15_6
```

This directory contains:

File	Purpose
`Containerfile`	Instructions for building the Postfix+OpenDKIM image
`docker-entrypoint.sh`	Script that launches both services inside container
`postinstall-dkim.sh`	Automated setup and build script
`README.md`	Short usage notes
`testing-guide.txt`	Reference testing commands

3.5 Run the Postinstall Script

Execute the build and environment configuration script:

```
sudo ./postinstall-dkim.sh eth0 xxx.xxx.xxx.0/24 xxx.xxx.xxx.254
```

Explanation of parameters:

Argument	Example	Purpose
`eth0`		Parent NIC for macvlan network
`xxx.xxx.xxx.0/24`		Subnet used by DKIM containers
`xxx.xxx.xxx.254`		Gateway IP on your LAN

This script:

- Installs Podman and its dependencies (netavark, aardvark-dns, swaks, etc.)
- Creates the macvlan network `dkimnet`
- Builds the container image `postfix-opendkim:leap15_6`
- Configures journald log rotation (max 500 MB)

3.6 Verify Podman Installation

After running the script, confirm installation:

```
podman --version
podman info | grep -E 'networkBackend|Version'
```

Expected example output:

```
Version: 5.4.2
networkBackend: netavark
```

Confirm that the macvlan network exists:

```
podman network ls
```

Sample output:

```
NETWORK ID    NAME      DRIVER    SUBNET          GATEWAY
a12b34c56d7e  dkimnet   macvlan   xxx.xxx.xxx.0/24 xxx.xxx.xxx.254
```

3.7 Verify the Container Image

List available container images:

```
podman images
```

Expected output:

```
REPOSITORY                  TAG         IMAGE ID       CREATED         SIZE
localhost/postfix-opendkim leap15_6   bb9200be5fec   5 minutes ago   650MB
```

If no image appears, rebuild manually:

```
podman build -t postfix-opendkim:leap15_6 -f
/var/Software/podman_dkim_bundle_leap15_6/Containerfile
```

3.8 Configure Persistent Logging

Ensure that journald and container logs are persistent:

```
sudo mkdir -p /var/log/journal
```

```
sudo systemctl restart systemd-journald
```

```
sudo journalctl --disk-usage
```

You should see:

```
Archived and active journals take up X MB on disk.
```

3.9 Create Base Directory Structure for Domains

Create top-level directories for each domain environment (you'll populate them later in Section 5):

```
sudo mkdir -p /srv/dkim/mydomain.com/{postfix,opendkim,logs}
```

Confirm structure:

```
tree -L 2 /srv/dkim
```

Expected:

```
/srv/dkim/
├── mydomain.com
│   ├── postfix
│   ├── opendkim
```

```
|   └── logs
├── mydomain.com
|   ├── postfix
|   ├── opendkim
|   └── logs
...
```

3.10 Verify Disk and Network Before Proceeding

Confirm adequate free space:

```
df -h /srv /var
```

Ensure each domain’s directory is writable:

```
sudo touch /srv/dkim/mydomain.com/testfile && ls -l /srv/dkim/mydomain.com/testfile
sudo rm /srv/dkim/mydomain.com/testfile
```

Check network connectivity one more time before container setup.

SECTION 4 – BUILDING THE CONTAINER ENVIRONMENT

This section describes how the DKIM container environment is constructed from the provided **Containerfile**, what each component does, and how to rebuild or modify it safely in the future.

4.1 Overview

Each DKIM relay runs inside a self-contained **Podman container** built from a custom image.
This image bundles the following software:

Package	Purpose
Postfix	Handles SMTP routing and relay through NO-IP
OpenDKIM	Performs DKIM signing of outbound mail
ca-certificates	Ensures valid TLS connections to NO-IP
findutils	Provides utilities required by Postfix setup routines

The image is built using a `Containerfile` located in `/var/Software/podman_dkim_bundle_leap15_6`.

4.2 Containerfile Explained

Open the file:

```
cat /var/Software/podman_dkim_bundle_leap15_6/Containerfile
```

It should look similar to the following:

```
# Containerfile – Postfix + OpenDKIM for openSUSE Leap 15.6
FROM registry.opensuse.org/opensuse/leap:15.6

RUN zypper --non-interactive refresh && \
    zypper --non-interactive install postfix opendkim ca-certificates findutils && \
    zypper --non-interactive clean && \
    (getent group postfix  || groupadd -r postfix)  && \
    (getent group postdrop || groupadd -r postdrop) && \
    (getent passwd postfix || useradd -r -s /sbin/nologin -g postfix postfix) && \
    postfix set-permissions || true

COPY docker-entrypoint.sh /usr/local/bin/docker-entrypoint.sh
RUN chmod +x /usr/local/bin/docker-entrypoint.sh

EXPOSE 25
ENTRYPOINT ["/usr/local/bin/docker-entrypoint.sh"]
```

4.3 Line-by-Line Breakdown

Line	Description
`FROM registry.opensuse.org/...`	Uses the official openSUSE Leap 15.6 image as the base.
`RUN zypper ... install`	Installs Postfix, OpenDKIM, and essential utilities.
`groupadd` / `useradd`	Ensures Postfix user and groups exist for proper permissions.
`postfix set-permissions`	Fixes directory and file permissions for Postfix spool files.
`COPY docker-entrypoint.sh`	Copies the entrypoint script into the image.
`RUN chmod +x`	Makes the script executable.
`EXPOSE 25`	Declares SMTP port 25 for clarity (container only; actual network IPs are macvlan).
`ENTRYPOINT`	Tells Podman to start the container using the entrypoint script.

4.4 Entrypoint Script

This script (`docker-entrypoint.sh`) controls startup of both services (OpenDKIM and Postfix).
View it with:

```
cat /var/Software/podman_dkim_bundle_leap15_6/docker-entrypoint.sh
```

Content:

```
#!/usr/bin/env bash
set -e
mkdir -p /run/opendkim /var/spool/postfix/public /var/spool/postfix/maildrop
chown opendkim:opendkim /run/opendkim || true
chgrp -R postdrop /var/spool/postfix/public /var/spool/postfix/maildrop || true
if [ -f /etc/opendkim/opendkim.conf ]; then
  /usr/sbin/opendkim -f -x /etc/opendkim/opendkim.conf &
fi
exec /usr/sbin/postfix start-fg
```

Explanation:

- Creates temporary runtime directories for OpenDKIM and Postfix queues.
- Ensures proper ownership of these directories.
- Starts OpenDKIM first in the background.
- Finally starts Postfix in the foreground (so Podman monitors the process).

4.5 Rebuilding the Image Manually

If you ever modify the Containerfile or entrypoint script, rebuild the image:

```
cd /var/Software/podman_dkim_bundle_leap15_6
sudo podman build -t postfix-opendkim:leap15_6 -f Containerfile
```

Check build success:

```
podman images
```

Expected result:

```
REPOSITORY                  TAG        IMAGE ID       CREATED         SIZE
localhost/postfix-opendkim  leap15_6   <unique_id>    <timestamp>     ~650MB
```

4.6 Verifying Runtime Environment

Run a temporary interactive container to verify Postfix and OpenDKIM versions:

```
sudo podman run --rm -it localhost/postfix-opendkim:leap15_6 bash
postfix -v
opendkim -V
exit
```

You should see outputs similar to:

```
postfix: mail_version = 3.6.x
OpenDKIM Filter v2.11.x
```

4.7 Understanding Persistent Volumes

The container mounts host directories at runtime for persistence.
Anything inside `/etc/postfix` and `/etc/opendkim` inside the container is linked to the corresponding `/srv/dkim/<domain>/` folders on the host.

Host Path	Container Path	Purpose
`/srv/dkim/<domain>/postfix`	`/etc/postfix`	Postfix configuration and credentials
`/srv/dkim/<domain>/opendkim`	`/etc/opendkim`	DKIM configuration and keys
`/srv/dkim/<domain>/logs`	`/var/log`	Persistent logs

4.8 Cleaning Up Old Builds

To remove older images or rebuilds:

```
sudo podman image prune -f
sudo podman system prune -f
```

This clears dangling images and cache layers.

4.9 Notes on Compatibility

- The image uses the **netavark** backend (Podman’s current default).

- The container can also be rebuilt on Leap 16 or SLES 15 SP6 with minimal modification.
- No root-level Podman service is required; each container runs standalone.

SECTION 5 – DOMAIN CONFIGURATION

This section defines how to configure Postfix and OpenDKIM for each domain so that each container can sign outbound messages correctly and relay through the NO-IP authenticated SMTP service.
Every domain operates independently, with its own configuration files, credentials, and signing keys.

5.1 Directory Structure

Each domain has its own configuration and log directories stored on the host under `/srv/dkim`.

Create the directories for all domains:

```
sudo mkdir -p /srv/dkim/{mydomain.com,mydomain.com,ezeasproducts.com.au,example.org}/{postfix,opendkim,logs}
```

```
sudo chown -R root:root /srv/dkim
```

```
sudo chmod -R 755 /srv/dkim
```

Resulting structure:

```
/srv/dkim/
├── mydomain.com/
│   ├── postfix/
│   ├── opendkim/
│   └── logs/
├── mydomain1.com/
│   ├── postfix/
│   ├── opendkim/
│   └── logs/
```

Each subdirectory is mounted into the container at runtime (see Section 7).

5.2 Postfix Configuration

Postfix handles outbound mail delivery and authentication with NO-IP.
Create the following files in `/srv/dkim/<domain>/postfix`.

5.2.1 main.cf

This is the main configuration file controlling relay settings, DKIM integration, and TLS.

Example for **mydomain.com** (`/srv/dkim/mydomain.com/postfix/main.cf`):

```
# Postfix main configuration for mydomain.com
```

```
myhostname = dkim.mydomain.com
```

```
mydomain   = mydomain.com
myorigin   = $mydomain

inet_interfaces = all
inet_protocols  = ipv4
mynetworks      = 127.0.0.0/8, 10.0.0.0/8

# --- DKIM Integration ---
smtpd_milters         = inet:127.0.0.1:8891
non_smtpd_milters     = inet:127.0.0.1:8891
milter_default_action = accept
milter_protocol       = 2

# --- Outbound Relay (NO-IP) ---
relayhost = [smtp-auth.no-ip.com]:465
smtp_sasl_auth_enable = yes
smtp_sasl_password_maps = hash:/etc/postfix/sasl_passwd
smtp_sasl_security_options = noanonymous
smtp_tls_security_level = encrypt
smtp_tls_CAfile = /etc/ssl/certs/ca-certificates.crt
smtp_tls_wrappermode = yes

relay_domains = $mydomain
compatibility_level = 3.6
```

Repeat for each domain, replacing the hostname and domain name appropriately.

To configure the relay for direct MX delivery, (operate as a normal MTA)

See Section Implement a normal outbound MTA (Mail Transfer Agent).

5.2.2 master.cf

This file defines Postfix service processes.
Start from the default image version:

```
sudo podman run --rm localhost/postfix-opendkim:leap15_6 \
```

```
cat /etc/postfix/master.cf > /srv/dkim/<domain>/postfix/master.cf
```

Typically, no changes are required unless you have special routing needs.

5.2.3 sasl_passwd

This file contains authentication credentials for the NO-IP relay service.

Example (`/srv/dkim/mydomain.com/postfix/sasl_passwd`):

```
[smtp-auth.no-ip.com]:465 mydomain.com@noip-smtp:YOUR_NOIP_PASSWORD
```

Set secure permissions:

```
sudo chmod 600 /srv/dkim/mydomain.com/postfix/sasl_passwd
```

After the container starts, this file must be compiled into a Postfix hash map:

```
sudo podman exec -it dkim-mydomain-com postmap /etc/postfix/sasl_passwd
```

This creates `/etc/postfix/sasl_passwd.db` inside the container.

5.3 OpenDKIM Configuration

OpenDKIM performs message signing using RSA keys.
Create the following files under `/srv/dkim/<domain>/opendkim`.

5.3.1 opendkim.conf

Defines OpenDKIM runtime behaviour and file paths.

Example `/srv/dkim/mydomain.com/opendkim/opendkim.conf`:

```
Syslog yes
UMask 002
UserID opendkim
Background yes
Socket inet:8891@localhost
PidFile /run/opendkim/opendkim.pid

Mode sv
Canonicalization relaxed/simple
LogWhy yes

KeyTable /etc/opendkim/KeyTable
SigningTable /etc/opendkim/SigningTable
InternalHosts /etc/opendkim/TrustedHosts
```

5.3.2 KeyTable

Maps selectors to key files.
Each selector corresponds to one DKIM key pair.

Example `/srv/dkim/mydomain.com/opendkim/KeyTable`:

```
eacb2025._domainkey.mydomain.com
mydomain.com:xxxx2025:/etc/opendkim/keys/mydomain.com/eacb2025.private
```

5.3.3 SigningTable

Defines which addresses are signed by which key.

Example `/srv/dkim/mydomain.com/opendkim/SigningTable`:

```
mydomain.com eacb2025._domainkey.mydomain.com
```

5.3.4 TrustedHosts

Specifies which clients' mail is to be signed (the DKIM relay signs everything from GWIA and localhost):

```
127.0.0.1

localhost

10.0.0.0/8

xxx.xxx.xxx.0/24
```

5.4 Generate DKIM Keys

Generate a unique DKIM key pair for each domain.
Run the following command (replace `<domain>` and `<selector>`):

1) Create the destination folder on the host

```
sudo mkdir -p /srv/dkim/mydomain.com/opendkim/keys/mydomain.com
```

2) Generate the key (override entrypoint so nothing else starts)

```
sudo podman run --rm \
  --entrypoint opendkim-genkey \
  -v /srv/dkim/mydomain.com/opendkim:/work:Z \
  localhost/postfix-opendkim:leap15_6 \
  -b 2048 -s xxxx2025 -d mydomain.com -D /work/keys/mydomain.com
```

3) Lock down the private key

```
sudo chmod 600 /srv/dkim/mydomain.com/opendkim/keys/mydomain.com/xxxx2025.private
```

You should now have:

```
/srv/dkim/mydomain.com/opendkim/keys/mydomain.com/xxxx2025.private
/srv/dkim/mydomain.com/opendkim/keys/mydomain.com/xxxx2025.txt
```

Wire into OpenDKIM tables (if not already)

```
# KeyTable
sudo tee /srv/dkim/mydomain.com/opendkim/KeyTable >/dev/null <<'EOF'
xxxx2025._domainkey.mydomain.com
mydomain.com:xxxx2025:/etc/opendkim/keys/mydomain.com/xxxx2025.private
EOF

# SigningTable
sudo tee /srv/dkim/mydomain.com/opendkim/SigningTable >/dev/null <<'EOF'
mydomain.com xxxx2025._domainkey.mydomain.com
EOF

# TrustedHosts (example)
sudo tee /srv/dkim/mydomain.com/opendkim/TrustedHosts >/dev/null <<'EOF'
127.0.0.1
localhost
10.0.0.0/8
EOF
```

Make sure your `/srv/dkim/mydomain.com/opendkim/opendkim.conf` is **table-based** (no `KeyFile` or `Selector` lines):

```
Syslog yes
UMask 002
UserID opendkim
Background yes
Socket inet:8891@localhost
PidFile /run/opendkim/opendkim.pid
Mode sv
Canonicalization relaxed/simple
LogWhy yes

KeyTable      /etc/opendkim/KeyTable
SigningTable  /etc/opendkim/SigningTable
```

```
InternalHosts /etc/opendkim/TrustedHosts
```

Restart that domain's container:

```
sudo podman restart dkim-mydomain-com  # use your actual container name
```

Publish DNS and validate

1. Copy the contents of:

```
cat /srv/dkim/mydomain.com/opendkim/keys/mydomain.com/xxxx2025.txt
```

Create a **TXT** record:

- **Name:** `xxxx2025._domainkey.mydomain.com`
- **Value:** the line from the `.txt` file (one line, quoted)

2. After DNS propagates, test:

```
sudo podman exec -it dkim-mydomain-com opendkim-testkey -d mydomain.com -s xxxx2025 -vvv
```

You should see **"key OK"** once the TXT is live.

Repeat for other domains (change domain + selector)

Example for mydomain.com:

```
sudo mkdir -p /srv/dkim/mydomain.com/opendkim/keys/mydomain.com
sudo podman run --rm --entrypoint opendkim-genkey \
  -v /srv/dkim/mydomain.com/opendkim:/work:Z \
  localhost/postfix-opendkim:leap15_6 \
  -b 2048 -s eacb2025 -d mydomain.com -D /work/keys/mydomain.com
# Set proper permissions
chmod 755 /etc/opendkim/
chmod 755 /etc/opendkim/keys/
chmod 700 /etc/opendkim/keys/mydomain.com/
chmod 600 /etc/opendkim/keys/mydomain.com/xxxx2025.private
sudo chmod 600 /srv/dkim/mydomain.com/opendkim/keys/mydomain.com/xxxx2025.private
chown -R opendkim:opendkim /etc/opendkim/keys/mydomain.com/
```

(Then update that domain's `KeyTable`, `SigningTable`, publish DNS, and restart the container.)

If anything else errors out, paste the exact command + one or two lines of output and I'll adjust the incantation.

Each command produces two files:

```
<selector>.private   (the private signing key)
```

```
<selector>.txt       (the DNS TXT record)
```

5.5 Publish the DKIM Public Key

Each `<selector>.txt` file contains a line like this:

```
eacb2025._domainkey IN TXT "v=DKIM1; k=rsa; p=MIIBIjANBgkqh..."
```

Add this as a TXT record in your domain's DNS:

Name	Type	Value
eacb2025._domainkey	TXT	"v=DKIM1; k=rsa; p=MIIBIjANBg..."

Verification:

```
dig TXT eacb2025._domainkey.mydomain.com +short
```

5.6 Permissions and Validation

Ensure all permissions are secure:

```
sudo chown -R root:root /srv/dkim
sudo chmod -R 755 /srv/dkim
sudo chmod 600 /srv/dkim/*/postfix/sasl_passwd
```

Validate structure:

```
ls -l /srv/dkim/mydomain.com/postfix/{main.cf,master.cf,sasl_passwd}
ls -l /srv/dkim/mydomain.com/opendkim/{opendkim.conf,KeyTable,SigningTable,TrustedHosts}
```

5.7 Repeat for Each Domain

Perform these steps for:

- mydomain.com (selector `eacb2025`)
- mydomain.com (selector `xxxx2025`)
- ezeasproducts.com.au (selector `ezeas2025`)
- example.org (selector `sample2025`)

SECTION 6 – GENERATING DKIM KEYS

This section details the complete process of generating, verifying, and maintaining DKIM keys for each domain configured in Section 5.
Although the key-generation commands were introduced previously, this section formalises the entire procedure and includes DNS validation and testing.

6.1 Purpose of DKIM Keys

Every domain that sends outbound mail must sign messages using a private key.
The corresponding **public key** is published in the DNS zone file for that domain.

Mail recipients (e.g. Gmail, Outlook, Yahoo) validate the message by comparing the signature in the email header with the published public key.
If they match, the message passes DKIM authentication.

6.2 Understanding the Terminology

Term	Meaning
Selector	A short label identifying which DKIM key is in use (e.g. `eacb2025`). It allows rotation without downtime.
Private Key (.private)	Stored on the DKIM host; used by OpenDKIM to sign messages.
Public Key (.txt)	Published as a TXT record in DNS; used by recipients to verify authenticity.
Key Directory	Location of key pairs under `/srv/dkim/<domain>/opendkim/keys/<domain>/`.

6.3 Key Generation Steps

You will perform these steps once for each domain.
Run the following command to generate a new 2048-bit key pair.

```
sudo podman run --rm -v /srv/dkim/<domain>/opendkim:/work:Z \
  localhost/postfix-opendkim:leap15_6 \
  bash -lc 'mkdir -p /work/keys/<domain> && \
  opendkim-genkey -b 2048 -s <selector> -d <domain> -D /work/keys/<domain> && \
  chmod 600 /work/keys/<domain>/<selector>.private'
```

Example for **mydomain.com**:

```
sudo podman run --rm -v /srv/dkim/mydomain.com/opendkim:/work:Z \
  localhost/postfix-opendkim:leap15_6 \
  bash -lc 'mkdir -p /work/keys/mydomain.com && \
  opendkim-genkey -b 2048 -s xxxx2025 -d mydomain.com -D /work/keys/mydomain.com && \
  chmod 600 /work/keys/mydomain.com/xxxx2025.private'
```

Resulting files:

`/srv/dkim/mydomain.com/opendkim/keys/mydomain.com/xxxx2025.private`

`/srv/dkim/mydomain.com/opendkim/keys/mydomain.com/xxxx2025.txt`

6.4 Publishing the Public Key

The `.txt` file contains a line similar to the following:

```
xxxx2025._domainkey IN TXT "v=DKIM1; k=rsa; p=MIIBIjANBgkqhkiG9w0B..."
```

Add this record to your DNS provider's control panel.
If your DNS interface separates *Name* and *Value* fields:

Field	Example
Name	`xxxx2025._domainkey`
Type	TXT
Value	`"v=DKIM1; k=rsa; p=MIIBIjANBgkqhkiG9w0B..."`

Wait 5–10 minutes for propagation, then test:

```
dig TXT xxxx2025._domainkey.mydomain.com +short
```

Expected output:

```
"v=DKIM1; k=rsa; p=MIIBIjANBgkqhkiG9w0B..."
```

If no record appears, check for quoting errors or missing underscores.

6.5 Verifying Key Integrity

Run a verification test from inside the container (once it's running):

```
sudo podman exec -it dkim-mydomain-com opendkim-testkey -d mydomain.com -s xxxx2025 -vvv
```

Expected output:

```
opendkim-testkey: key OK
```

Common issues:

Problem	Resolution
"key not secure"	DNS propagation not complete; wait 30 minutes and retry
"no key"	TXT record not published correctly
"permission denied"	Incorrect file permissions; re-run `chmod 600` on the private key

---

6.6 Key Rotation (Yearly Recommended)

To rotate a key without downtime:

1. Generate a new selector (e.g. `eacb2026`).
2. Add its DNS TXT record in parallel with the old one.

3. Update the `KeyTable` and `SigningTable` for that domain.
4. Reload OpenDKIM (`sudo podman exec -it dkim-<domain> pkill opendkim`).
5. Remove the old key after a week of confirmed success.

6.7 Backup of Key Material

Because DKIM relies on cryptographic keys, **always back up** the entire `/srv/dkim/<domain>/opendkim/keys/` directory.
A simple weekly cron job can tarball all keys:

```
sudo tar czf /backup/dkim-keys-$(date +%F).tar.gz /srv/dkim/*/opendkim/keys
```

Keep this archive in a secure, access-restricted location.

SECTION 7 – RUNNING CONTAINERS

This section details how to launch, verify, and manage the DKIM containers for each domain.
Each container runs its own Postfix + OpenDKIM stack, has a static IP address, and mounts its configuration directories from `/srv/dkim/<domain>`.

7.1 Understanding the Container Run Command

Each DKIM relay container is created using the `podman run` command.
The basic syntax is:

```
sudo podman run -d --name dkim-<domain> \
  --network dkimnet --ip <ip-address> \
  --tmpfs /run/opendkim:mode=0755 \
  -v /srv/dkim/<domain>/opendkim:/etc/opendkim:Z \
  -v /srv/dkim/<domain>/postfix:/etc/postfix:Z \
  -v /srv/dkim/mydomain.com/ssl:/etc/postfix/ssl:Z \
  -v /srv/dkim/<domain>/logs:/var/log:Z \
  localhost/postfix-opendkim:leap15_6
```

Explanation of parameters:

Option	Description
`-d`	Detached mode (runs in the background)
`--name`	Unique container name for easy management
`--network dkimnet`	Connects container to the shared macvlan network
`--ip <ip>`	Assigns a unique static IP to this container
`--tmpfs /run/opendkim`	Creates temporary directory for OpenDKIM runtime files
`-v`	Mounts host directories for configuration persistence
`localhost/postfix-opendkim:leap15_6`	The container image built earlier

7.2 Create Containers for Each Domain

mydomain.com

```
sudo podman run -d --name dkim-mydomain-com \
--network dkimnet --ip xxx.xxx.xxx.xxx \
--tmpfs /run/opendkim:mode=0755 \
-v /srv/dkim/mydomain.com/opendkim:/etc/opendkim:Z \
-v /srv/dkim/mydomain.com/postfix:/etc/postfix:Z \
-v /srv/dkim/mydomain.com/ssl:/etc/postfix/ssl:Z \
-v /srv/dkim/mydomain.com/logs:/var/log:Z \
```

```
-e TZ=Australia/Brisbane \
localhost/postfix-opendkim:leap15_6
```

mtamydomain.com (Direct MTA DKIM Relay)

```
sudo podman run -d --name dkim-mtamydomain-com \
--network dkimnet --ip xxx.xxx.xxx.159 \
--tmpfs /run/opendkim:mode=0755 \
-v /srv/dkim/mtamydomain.com/opendkim:/etc/opendkim:Z \
-v /srv/dkim/mtamydomain.com/postfix:/etc/postfix:Z \
-v /srv/dkim/mtamydomain.com/ssl:/etc/postfix/ssl:Z \
-v /srv/dkim/mtamydomain.com/logs:/var/log:Z \
-e TZ=Australia/Brisbane \
localhost/postfix-opendkim:leap15_6
```

example.org (Generic Example)

```
sudo podman run -d --name dkim-example-org \
  --network dkimnet --ip xxx.xxx.xxx.133 \
  --tmpfs /run/opendkim:mode=0755 \
  -v /srv/dkim/example.org/opendkim:/etc/opendkim:Z \
  -v /srv/dkim/example.org/postfix:/etc/postfix:Z \
  -v /srv/dkim/example.org/logs:/var/log:Z \
  -e TZ=Australia/Brisbane \
localhost/postfix-opendkim:leap15_6
```

7.3 Initialise SASL Password Database

Once each container is running, compile the SASL password map inside it:

```
sudo podman exec -it dkim-mydomain-com postmap /etc/postfix/sasl_passwd
```

Verify the database exists:

```
sudo podman exec -it dkim-mydomain-com ls /etc/postfix/sasl_passwd.lmdb
```

Expected output:

```
/etc/postfix/sasl_passwd.db
```

7.4 Confirm Containers Are Running

List all active containers:

```
podman ps
```

Expected output example:

```
CONTAINER ID  IMAGE                                    STATUS         NAMES
e2a11b22c33d  localhost/postfix-opendkim:leap15_6  Up 10 minutes  dkim-mydomain-com
c3d44e55f66g  localhost/postfix-opendkim:leap15_6  Up 10 minutes  dkim-mydomain-com
...
```

7.5 Check Logs for Startup Messages

View the logs for a container:

```
sudo podman logs --tail=50 dkim-mydomain-com
```

You should see:

```
postfix/postlog: starting the Postfix mail system
OpenDKIM Filter v2.11.x starting
```

If OpenDKIM reports:

```
KeyFile and Selector must both be defined or both be undefined
```

→ Check that `KeyFile` is **not** listed in `opendkim.conf` (only use KeyTable/SigningTable).

7.6 Testing Service Status

Enter a container shell to confirm services:

```
sudo podman exec -it dkim-mydomain-com bash
postfix status
pgrep opendkim && echo "OpenDKIM running"
exit
```

Expected:

```
postfix/postfix-script: the Postfix mail system is running
OpenDKIM running
```

7.7 Restart and Stop Commands

Action	Command
Restart container	`sudo podman restart dkim-mydomain-com`
Stop container	`sudo podman stop dkim-mydomain-com`
Start container	`sudo podman start dkim-mydomain-com`
Remove container	`sudo podman rm -f dkim-mydomain-com`

7.8 Automatic Startup (Optional)

You can generate a systemd unit file for automatic container start at boot:

```
sudo podman generate systemd --name dkim-mydomain-com --files --new
sudo mv container-dkim-mydomain-com.service /etc/systemd/system/
sudo systemctl enable --now container-dkim-mydomain-com
```

Repeat for each container as needed.

7.9 Quick Verification Checklist

Check	Command	Expected Result
Containers running	`podman ps`	All DKIM containers listed
Network active	`podman network inspect dkimnet`	Subnet and IPs match configuration
Postfix logs	`podman logs dkim-<domain>`	“Postfix mail system is running”
OpenDKIM running	`pgrep opendkim` (inside container)	PID returned
Relay connection	`openssl s_client -connect smtp-auth.no-ip.com:465`	Valid certificate

SECTION 8 – TESTING AND VERIFICATION

This section verifies that each DKIM container signs messages correctly and relays them securely through NO-IP to external recipients.
All tests can be performed before GroupWise is connected to the relay.

8.1 Confirm Outbound Connectivity

First, ensure the container can reach NO-IP over port 465:

```
sudo podman exec -it dkim-mydomain-com openssl s_client -connect smtp-auth.no-ip.com:465 -crlf -quiet
```

Expected output:

```
CONNECTED(00000003)
```

```
220 smtp-auth.no-ip.com ESMTP NOIP SMTP Relay
```

Type `QUIT` and press **Enter** to exit the session.

If connection fails, verify:

- Firewall rules permit outbound TCP 465.
- DNS resolution for `smtp-auth.no-ip.com` works.
- The relay host credentials are correct.

8.2 Send a Test Message Using swaks

Use `swaks` (installed during setup) to send a test message from each domain.
Example for **mydomain.com**:

```
swaks --from test@mydomain.com --to youraddress@gmail.com \
  --server xxx.xxx.xxx.xxx --tls \
  --auth \
  --header "Subject: DKIM Test mydomain.com"
```

Repeat for each container by changing the server IP and sender domain.

If you receive “Mail accepted for delivery”, the relay is functioning.

8.3 Checking Headers in Gmail or Outlook

Open the received message, then:

- **Gmail:** click “⋮ > Show Original”
- **Outlook Web:** click “View Message Source”

Look for lines similar to:

```
Authentication-Results: mx.google.com;
       dkim=pass header.i=@mydomain.com header.s=eacb2025 header.b=FAbc...
```

✅ **dkim=pass** means signing is successful.
If you see **dkim=fail**, check:

- The selector in DNS matches the one used in `KeyTable`.
- The TXT record is quoted correctly (no line breaks).
- OpenDKIM is using the right key path.

8.4 Verifying DNS Resolution from Container

Confirm the container can resolve the DKIM TXT record:

```
sudo podman exec -it dkim-mydomain-com dig TXT eacb2025._domainkey.mydomain.com +short
```

Expected:

```
"v=DKIM1; k=rsa; p=MIIBIjANBgkqhkiG9w0B..."
```

If blank, check your external DNS or wait for propagation.

8.5 Using opendkim-testkey

Validate that OpenDKIM can read and verify the key:

```
sudo podman exec -it dkim-mydomain-com opendkim-testkey -d mydomain.com -s eacb2025 -vvv
```

Expected output:

```
opendkim-testkey: key OK
```

Common results:

Message	Meaning	Action
`key OK`	All correct	None
`key not secure`	DNS still propagating	Wait and retry
`no key`	DNS TXT not found	Correct DNS entry
`permission denied`	Wrong key file perms	Run `chmod 600`

8.6 Testing Relay Authentication

From inside container:

```
sudo podman exec -it dkim-mydomain-com postconf relayhost smtp_sasl_password_maps
```

Should output:

```
relayhost = [smtp-auth.no-ip.com]:465
```

```
smtp_sasl_password_maps = hash:/etc/postfix/sasl_passwd
```

Then send a live relay test:

```
swaks --from test@mydomain.com --to test@outlook.com \
  --server xxx.xxx.xxx.xxx --tls
```

Verify in Outlook headers that mail passes DKIM and SPF (if configured).

8.7 Troubleshooting Failures

Symptom	Possible Cause	Resolution
`Connection timed out`	Firewall blocking port 465	Allow outbound 465
`Authentication failed`	Wrong credentials	Edit `sasl_passwd`, rerun `postmap`
`dkim=neutral`	DNS TXT missing	Verify DNS record
`postfix/postlog: not owned by group postdrop`	Permissions	Run `postfix set-permissions` inside container
`KeyFile and Selector both defined`	Legacy config	Remove KeyFile/Selector from `opendkim.conf`

8.8 Integrating with GroupWise

Once individual tests succeed, reconfigure GroupWise GWIA to route all external mail through the DKIM relay:

1. Log in to GroupWise Administration Console.
2. Navigate to **Internet Agent** > **Access Control**.
3. Set **Relay Host for Outbound Messages** to:

   `ebvl-dkimhost-001.cdn.mydomain.com`
4. Restart GWIA service.
5. Send test email from GroupWise to Gmail and confirm DKIM = PASS.

8.9 Monitoring Logs During Testing

Watch container logs in real time:

```
sudo podman logs -f dkim-mydomain-com
```

Typical successful startup sequence:

```
postfix/postlog: starting the Postfix mail system
OpenDKIM Filter v2.11.x starting
```

To tail specific logs inside the container:

```
sudo podman exec -it dkim-mydomain-com tail -f /var/log/maillog
```

8.10 Confirm End-to-End Functionality

When all the following conditions are met, the deployment is complete:

Verification Step	Expected Result
DNS TXT published	dig +short shows DKIM record
Container connectivity	openssl s_client connects successfully
swaks message delivered	Message accepted by NO-IP
Gmail/Outlook header	DKIM = PASS
No errors in logs	Clean postfix and opendkim output

SECTION 9 – MAINTENANCE AND ROTATION

This section describes ongoing maintenance procedures to ensure continued DKIM signing, relay reliability, and container health.
It covers updating NO-IP credentials, rotating DKIM keys, managing logs, and applying OS or image updates.

9.1 Regular Maintenance Overview

Task	Frequency	Description
Check DKIM relay logs	Weekly	Ensure signing and delivery are successful
Verify DNS TXT records	Quarterly	Confirm keys still resolve correctly
Rotate DKIM keys	Annually	Maintain cryptographic hygiene
Update NO-IP credentials	As needed	Renew credentials if rotated by NO-IP
Update container image	Quarterly	Rebuild image from Containerfile
Backup configuration	Weekly	Archive `/srv/dkim` and `/var/Software`

9.2 Checking DKIM and Relay Logs

View recent activity for a container:

```
sudo podman logs --tail=100 dkim-mydomain-com
```

Example normal messages:

```
postfix/postlog: starting the Postfix mail system
opendkim[123]: DKIM-Signature field added
```

To view logs inside container:

```
sudo podman exec -it dkim-mydomain-com tail -f /var/log/maillog
```

Search for signing or relay errors:

```
sudo podman exec -it dkim-mydomain-com grep -i error /var/log/maillog
```

9.3 Updating NO-IP Credentials

If NO-IP changes your relay password:

1. Edit `/srv/dkim/<domain>/postfix/sasl_passwd` and update the password line:

   ```
   [smtp-auth.no-ip.com]:465 <domain>@noip-smtp:NEWPASSWORD
   ```

2. Rebuild the map:

   ```
   sudo podman exec -it dkim-<domain> postmap /etc/postfix/sasl_passwd
   ```

3. Restart Postfix:

   ```
   sudo podman exec -it dkim-<domain> postfix reload
   ```

Check logs for:

```
SASL authentication succeeded
```

9.4 Rotating DKIM Keys

It’s good practice to rotate keys annually or if compromised.

1. Generate a new key pair with a new selector:

   ```
   sudo podman run --rm -v /srv/dkim/<domain>/opendkim:/work:Z \
     localhost/postfix-opendkim:leap15_6 \
     bash -lc 'mkdir -p /work/keys/<domain> && \
     opendkim-genkey -b 2048 -s <new_selector> -d <domain> -D /work/keys/<domain> && \
     chmod 600 /work/keys/<domain>/<new_selector>.private'
   ```

2. Add new entry to `KeyTable`:

   ```
   <new_selector>._domainkey.<domain>
   <domain>:<new_selector>:/etc/opendkim/keys/<domain>/<new_selector>.private
   ```

3. Add to `SigningTable`:

   ```
   *@<domain> <new_selector>._domainkey.<domain>
   ```

4. Publish new TXT record in DNS.

5. Restart OpenDKIM inside container:

   ```
   sudo podman exec -it dkim-<domain> pkill opendkim
   ```

   Podman automatically restarts it via entrypoint script.

6. After confirmation of “DKIM=PASS” in mail headers, remove old key entries.

9.5 Applying OS or Podman Updates

Update host packages periodically:

```
sudo zypper refresh && sudo zypper update -y
```

If the container image needs updates (e.g. newer Postfix/OpenDKIM):

```
cd /var/Software/podman_dkim_bundle_leap15_6
sudo podman build -t postfix-opendkim:leap15_6 -f Containerfile
sudo podman restart dkim-<domain>
```

Verify version changes inside container:

```
sudo podman exec -it dkim-<domain> postfix -v
sudo podman exec -it dkim-<domain> opendkim -V
```

9.6 Cleaning Log Files

Logs are stored under `/srv/dkim/<domain>/logs`.
Rotate or clear them monthly to prevent disk bloat.

Example log cleanup script (`/etc/cron.monthly/dkim-logrotate.sh`):

```
#!/usr/bin/env bash
find /srv/dkim/*/logs -type f -mtime +90 -delete
```

Make it executable:

```
sudo chmod +x /etc/cron.monthly/dkim-logrotate.sh
```

9.7 Checking Disk Usage

Monitor free space regularly:

```
df -h /srv /var
```

If `/srv` grows too large, compress old logs or relocate to a larger volume.

9.8 Validating Configuration Consistency

To compare all active domains quickly:

```
grep -H "relayhost" /srv/dkim/*/postfix/main.cf
grep -H "KeyTable" /srv/dkim/*/opendkim/opendkim.conf
```

All should return consistent results pointing to `smtp-auth.no-ip.com` and valid paths.

9.9 Testing Email Signatures After Maintenance

Always re-test DKIM functionality after key rotation, relay updates, or container rebuilds:

```
swaks --from admin@<domain> --to test@gmail.com --server <container-ip> --tls
```

Check for:

```
Authentication-Results: dkim=pass header.i=@<domain>
```

9.10 Quarterly Verification Checklist

Task	Completed
[ ] DKIM keys verified via opendkim-testkey	
[ ] DNS TXT records resolve correctly	
[ ] NO-IP credentials current	
[ ] Containers running and reachable	
[ ] Log directories within size limits	
[ ] Latest image built and running	

SECTION 10 – REBUILD FROM SCRATCH QUICK REFERENCE

This section is designed for rapid disaster recovery or VM rebuilds.
Follow these steps sequentially to fully restore a DKIM host and all containers.

10.1 Overview

Use this section when:

- The DKIM host VM was lost or replaced.
- You are migrating to new hardware or a new version of openSUSE.
- You need to completely re-deploy after a major incident.

Approximate time to recovery: **20–30 minutes**
Dependencies:

- Latest backup tarballs (`/srv/dkim`, `/var/Software/podman_dkim_bundle_leap15_6`)
- Access to NO-IP credentials and DNS management portal.

10.2 Rebuild Procedure

Step	Action	Command / Notes
1	Install openSUSE Leap 15.6	Choose minimal server install.
2	Update OS and install tools	`zypper refresh && zypper update -y`
3	Restore backups	`tar xzf /backup/dkim-backup-YYYY-MM-DD.tar.gz -C /`
4	Verify structure	`ls /srv/dkim/*/{postfix,opendkim,logs}`
5	Reinstall Podman bundle	`cd /var/Software/podman_dkim_bundle_leap15_6 && ./postinstall-dkim.sh eth0 xxx.xxx.xxx.0/24 xxx.xxx.xxx.254`
6	Rebuild container image	`podman build -t postfix-opendkim:leap15_6 -f Containerfile`
7	Recreate containers	(see Section 7 for full `podman run` commands)
8	Regenerate SASL maps	`podman exec -it dkim-<domain> postmap /etc/postfix/sasl_passwd`
9	Test connectivity	`openssl s_client -connect smtp-auth.no-ip.com:465`
10	Send test email	`swaks --from test@<domain> --to you@gmail.com --server <container-ip> --tls`
11	Confirm DKIM PASS	Check headers in Gmail or Outlook.
12	Enable auto-start (optional)	`podman generate systemd --name dkim-<domain> --files --new` then enable unit.

10.3 Minimum Verification Before Going Live

1. All containers listed in `podman ps`
2. `postfix` and `opendkim` processes active inside each container
3. Outbound connection to NO-IP successful (port 465)
4. DNS TXT record resolves (`dig TXT <selector>._domainkey.<domain> +short`)
5. `DKIM=PASS` result in received message headers

10.4 Time-Saving Tips

- Keep `/backup/dkim-latest.tar.gz` updated weekly with:

  ```
  tar czf /backup/dkim-latest.tar.gz /srv/dkim
  /var/Software/podman_dkim_bundle_leap15_6
  ```

- Maintain a plain-text note with:
  - Container IPs
  - Domain selectors
  - NO-IP usernames and relay passwords
- Store these securely in your password manager or encrypted vault.
- Test a single domain container first before re-enabling GroupWise relay.

10.5 Example Recovery Command Set

```
# Rebuild environment
zypper refresh && zypper update -y
tar xzf /backup/dkim-backup.tar.gz -C /
cd /var/Software/podman_dkim_bundle_leap15_6
./postinstall-dkim.sh eth0 xxx.xxx.xxx.0/24 xxx.xxx.xxx.254

# Rebuild image and containers
podman build -t postfix-opendkim:leap15_6 -f Containerfile
podman run -d --name dkim-mydomain-com \
  --network dkimnet --ip xxx.xxx.xxx.xxx \
  --tmpfs /run/opendkim:mode=0755 \
  -v /srv/dkim/mydomain.com/opendkim:/etc/opendkim:Z \
  -v /srv/dkim/mydomain.com/postfix:/etc/postfix:Z \
  -v /srv/dkim/mydomain.com/logs:/var/log:Z \
```

```
  localhost/postfix-opendkim:leap15_6
podman exec -it dkim-mydomain-com postmap /etc/postfix/sasl_passwd


# Validate and test
swaks --from test@mydomain.com --to you@gmail.com --server xxx.xxx.xxx.xxx --tls
```

Expected:
✅ “Message accepted for delivery”
✅ “DKIM=PASS” in headers.

10.6 When to Use This Section

- Disaster recovery scenario
- Hardware or host migration
- OS reinstallation
- Annual environment rebuild and audit

10.7 Completion Checklist

Step	Description	Done
1	VM installed and updated	[ ]
2	Backup restored	[ ]
3	Podman and network configured	[ ]
4	Containers rebuilt and running	[ ]
5	DKIM signing validated	[ ]
6	NO-IP relay authenticated	[ ]
7	GroupWise re-pointed to DKIM host	[ ]

APPENDIX A – COMMAND REFERENCE

This appendix lists the essential commands used during installation, operation, and maintenance of the DKIM Podman environment.
Use it as a quick-access cheat sheet when administering or rebuilding the system.

A.1 System Administration Commands

Task	Command	Description
Update OS	`sudo zypper refresh && sudo zypper update -y`	Refresh repository and apply updates.
Install base tools	`sudo zypper install -y vim htop iotop ncdu net-tools bind-utils curl wget git`	Useful admin tools.
Check disk usage	`df -h /srv /var`	Verify free space before or after deployment.
View logs (journald)	`sudo journalctl -xe`	Show recent system logs.
Restart journald	`sudo systemctl restart systemd-journald`	Ensure persistent logging is active.

A.2 Podman Commands

Task	Command	Description
Verify Podman version	`podman --version`	Confirm Podman installed.
Build container image	`podman build -t postfix-opendkim:leap15_6 -f Containerfile`	Build the DKIM relay image.
List available images	`podman images`	Check existing container images.
Run container	`podman run -d --name dkim-<domain> --network dkimnet --ip <ip> ...`	Start container in background.
List running containers	`podman ps`	Display active containers.
View container logs	`podman logs dkim-<domain>`	Show startup and mail processing logs.
Enter container shell	`podman exec -it dkim-<domain> bash`	Open an interactive shell.
Stop container	`podman stop dkim-<domain>`	Gracefully stop container.
Restart container	`podman restart dkim-<domain>`	Restart services quickly.
Remove container	`podman rm -f dkim-<domain>`	Delete container.
Clean old images	`podman image prune -f`	Remove unused images.
Generate systemd unit	`podman generate systemd --name dkim-<domain> --files --new`	Create service for auto-start.

A.3 Network and Connectivity

Task	Command	Description
Check network list	`podman network ls`	Verify dkimnet macvlan created.
Inspect network	`podman network inspect dkimnet`	Show IP and gateway details.
Test relay port	`openssl s_client -connect smtp-auth.no-ip.com:465`	Verify SSL/TLS connectivity.
DNS lookup test	`dig smtp-auth.no-ip.com +short`	Confirm DNS resolution.
Ping gateway	`ping -c3 xxx.xxx.xxx.254`	Confirm basic network reachability.

A.4 Postfix Management

Task	Command	Description
Reload Postfix	`podman exec -it dkim-<domain> postfix reload`	Apply configuration changes.
Restart Postfix	`podman exec -it dkim-<domain> postfix stop && postfix start`	Restart mail service.
Check Postfix version	`podman exec -it dkim-<domain> postfix -v`	Display version info.
Create SASL password map	`podman exec -it dkim-<domain> postmap /etc/postfix/sasl_passwd`	Compile SASL credentials.
Verify relay settings	`podman exec -it dkim-<domain> postconf relayhost`	Confirm NO-IP relay configuration.

A.5 OpenDKIM Management

Task	Command	Description
Restart OpenDKIM	`podman exec -it dkim-<domain> pkill opendkim`	Restarts signing service.
Verify key	`podman exec -it dkim-<domain> opendkim-testkey -d <domain> -s <selector> -vvv`	Check DNS key validity.
Check OpenDKIM version	`podman exec -it dkim-<domain> opendkim -V`	Confirm software version.

A.6 Testing and Verification

Task	Command	Description
Send test message	`swaks --from test@<domain> --to you@gmail.com --server <container-ip> --tls`	Test mail delivery and signing.
View DKIM result	(Gmail: “Show Original”)	Confirm `DKIM=PASS` header.
View relay logs	`podman logs --tail=50 dkim-<domain>`	Display recent activity.
DNS TXT test	`dig TXT <selector>._domainkey.<domain> +short`	Verify public DKIM

Task	Command	Description
		record.

A.7 Backup and Restore

Task	Command	Description
Backup configuration	`tar czf /backup/dkim-$(date +%F).tar.gz /srv/dkim /var/Software`	Full daily/weekly backup.
Restore from backup	`tar xzf /backup/dkim-backup.tar.gz -C /`	Restore directories.
Verify backup integrity	`tar -tzf /backup/dkim-YYYY-MM-DD.tar.gz | head`	

APPENDIX B – DNS TXT TEMPLATES FOR ALL DOMAINS

DKIM requires a **TXT record** published in DNS under
`<selector>._domainkey.<domain>`
for each domain that sends outbound mail.

This appendix lists the expected record format, examples for all deployed domains, and verification commands.

B.1 TXT Record Format

Field	Example	Description
Record Name	`eacb2025._domainkey.mydomain.com`	Selector + domain label
Record Type	TXT	Always “TXT” for DKIM
Record Value	`"v=DKIM1; k=rsa; p=<base64_public_key>"`	The public RSA key value generated by `opendkim-genkey`

Important notes:

- The key value (`p=`) must be **one continuous line** of base64 text.
- Wrap the value in quotes `"..."`.
- Do **not** include the “IN TXT” portion in most DNS panels — just the Name and Value fields.
- DNS updates can take up to 30 minutes to propagate globally.

B.2 Example – mydomain.com

Selector: `eacb2025`
DNS Record:

Field	Example
Name	`eacb2025._domainkey.mydomain.com`
Type	TXT
Value	`"v=DKIM1; k=rsa; p=MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8A...IDAQAB"`

Verification:

```
dig TXT eacb2025._domainkey.mydomain.com +short
```

Expected output:

```
"v=DKIM1; k=rsa; p=MIIBIjANBgkqhkiG9w0B..."
```

B.3 Example – mydomain.com

Selector: `xxxx2025`
DNS Record:

Field	Example
Name	`xxxx2025._domainkey.mydomain.com`
Type	TXT
Value	`"v=DKIM1; k=rsa; p=MIICIjANBgkqhkiG9w0BAQEFAAOCAQ8A...IDAQAB"`

Verification:

```
dig TXT xxxx2025._domainkey.mydomain.com +short
```

B.4 Example – ezeasproducts.com.au

Selector: `ezeas2025`
DNS Record:

Field	Example
Name	`ezeas2025._domainkey.ezeasproducts.com.au`
Type	TXT
Value	`"v=DKIM1; k=rsa; p=MIICIjANBgkqhkiG9w0BAQEFAAOCAQ8A...IDAQAB"`

Verification:

```
dig TXT ezeas2025._domainkey.ezeasproducts.com.au +short
```

B.5 Example – mydomain.com

Selector: `xxxx2025`
DNS Record:

Field	Example
Name	`xxxx2025._domainkey.mydomain.com`
Type	TXT
Value	`"v=DKIM1; k=rsa; p=MIICIjANBgkqhkiG9w0BAQEFAAOCAQ8A...IDAQAB"`

Verification:

```
dig TXT xxxx2025._domainkey.mydomain.com +short
```

B.6 Generic Template – example.org

Selector: `sample2025`
DNS Record:

Field	Example
Name	`sample2025._domainkey.example.org`
Type	TXT
Value	`"v=DKIM1; k=rsa; p=MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8A...IDAQAB"`

Verification:

```
dig TXT sample2025._domainkey.example.org +short
```

B.7 Common DNS Errors and Fixes

Issue	Cause	Resolution
TXT record not visible	DNS propagation delay	Wait up to 30 mins, verify with multiple resolvers
Line breaks in key	Copy/paste inserted newline	Paste value as one continuous line
Record quoted incorrectly	Extra quotes in control panel	Enter only single set of double quotes
Missing underscores	Typo in record name	Ensure record name is `<selector>._domainkey.<domain>`

APPENDIX C – TROUBLESHOOTING GUIDE

This appendix provides a structured reference for diagnosing and resolving common issues with the DKIM Podman relay environment, covering **OpenDKIM**, **Postfix**, **Podman**, **network**, and **DNS** problems.

C.1 How to Approach a Failure

1. **Identify where it failed:**
   - Build stage (image/container creation)
   - Startup (services failing)
   - Runtime (relay not signing, messages rejected)
2. **Use the logs:**
   - Container logs:

     ```
     podman logs dkim-<domain>
     ```
   - Live log inside container:

     ```
     podman exec -it dkim-<domain> tail -f /var/log/maillog
     ```
3. **Work from bottom up:**
   DNS → Network → Container → OpenDKIM → Postfix → NO-IP Relay → Recipient

---

C.2 OpenDKIM Errors

Error Message	Description	Solution
`KeyFile and Selector must both be defined or both be undefined`	The `KeyFile` directive is still present in `opendkim.conf`.	Remove any `KeyFile` or `Selector` lines; use `KeyTable` and `SigningTable` only.
`/run/opendkim: chdir(): No such file or directory`	Runtime directory not mounted or created.	Ensure container is started with `--tmpfs /run/opendkim:mode=0755`.
`permission denied`	Key file not readable by OpenDKIM.	`chmod 600 /srv/dkim/<domain>/opendkim/keys/<domain>/*.private`
`opendkim-testkey: key not secure`	DNS TXT not fully propagated.	Wait 30 minutes or verify via `dig`.
`no key found`	DNS TXT record missing.	Check correct selector and domain.
`OpenDKIM Filter not starting`	Wrong socket or user permissions.	Check `Socket inet:8891@localhost` and `UserID opendkim` in config.

C.3 Postfix Errors

Error Message	Description	Solution
`unknown group name: postdrop`	Postfix group missing in container.	Run inside container: `groupadd -r postdrop && postfix set-permissions`.
`not owned by group postdrop`	Incorrect permissions.	Run `postfix set-permissions`.
`no /etc/postfix/master.cf file found`	Missing base config.	Copy default from image: `podman run --rm localhost/postfix-opendkim:leap15_6 cat /etc/postfix/master.cf > /srv/dkim/<domain>/postfix/master.cf`.
`SASL authentication failed`	Incorrect relay credentials or map not compiled.	Edit `/srv/dkim/<domain>/postfix/sasl_passwd`, run `postmap` inside container.
`relay access denied`	`relayhost` missing or domain mismatch.	Confirm correct `relayhost` in `main.cf`.
`smtp_use_tls will be removed`	Deprecated directive warning.	Safe to ignore; modern setting is `smtp_tls_security_level = encrypt`.

C.4 Podman / Container Issues

Symptom	Cause	Fix
`podman: the specified Containerfile does not exist`	Wrong path or missing file.	Run from `/var/Software/podman_dkim_bundle_leap15_6`.
`Unit dkim-<domain>.container.service does not exist`	Quadlet not supported on Leap 15.6.	Use `podman run` manually or generate systemd units (see Section 7.8).
`container exits immediately`	Entry script crash (missing config).	Check that `/srv/dkim/<domain>/postfix/main.cf` and `opendkim.conf` exist.
`podman network not found`	Network not created.	`podman network create --driver macvlan --subnet xxx.xxx.xxx.0/24 --gateway xxx.xxx.xxx.254 -o parent=eth0 dkimnet`
`permission denied creating /run/opendkim`	No tmpfs for /run directory.	Add `--tmpfs /run/opendkim:mode=0755` to `podman run` command.

C.5 Network and DNS Problems

Issue	Likely Cause	Fix
`Connection timed out` on port 465	Firewall blocking outbound SSL/TLS.	Allow outbound TCP 465 in `firewalld` or `nftables`.
Cannot ping NO-IP host	Missing DNS or gateway.	Check `/etc/resolv.conf`, confirm `ping smtp-auth.no-ip.com`.

Issue	Likely Cause	Fix
DNS query returns nothing	Typo or propagation delay.	Wait or verify correct selector and record name.
Incorrect PTR (reverse DNS)	Internal-only network.	Usually safe to ignore for outbound relay.

C.6 Testing Failures

Symptom	Likely Cause	Resolution
Email not arriving at recipient	Relay authentication failure or blocked by NO-IP.	Check `maillog` for SASL or connection errors.
DKIM shows `neutral`	Key mismatch or missing header signing.	Verify DNS and `KeyTable`.
Gmail shows “no DKIM”	Message relayed bypassing filter.	Ensure GWIA points to DKIM relay host only.
`postfix: connect to 127.0.0.1:8891 refused`	OpenDKIM service not running.	Restart container or check entrypoint logs.
Multiple duplicate emails	GroupWise retrying failed relay.	Confirm DKIM relay accepts outbound and returns 250 status.

C.7 Diagnostic Commands Reference

Purpose	Command
Check all running containers	`podman ps`
Show logs for specific container	`podman logs dkim-<domain>`
Tail logs inside container	`podman exec -it dkim-<domain> tail -f /var/log/maillog`
Test relay host connection	`openssl s_client -connect smtp-auth.no-ip.com:465`
Check DKIM DNS record	`dig TXT <selector>._domainkey.<domain> +short`
Verify key functionality	`podman exec -it dkim-<domain> opendkim-testkey -d <domain> -s <selector> -vvv`
View Postfix queue	`podman exec -it dkim-<domain> mailq`

C.8 Escalation Procedure

If a container consistently fails:

1. Stop and remove it:

   ```
   podman stop dkim-<domain> && podman rm dkim-<domain>
   ```

2. Recreate it with the original `podman run` command.

3. If issue persists, rebuild image:

   ```
   podman build -t postfix-opendkim:leap15_6 -f Containerfile
   ```

4. If still unresolved:
    - Verify host DNS
    - Recreate network `dkimnet`
    - Reboot host

C.9 Common Success Indicators

Indicator	Meaning
`postfix/postlog: starting the Postfix mail system`	Relay successfully started.
`OpenDKIM Filter v2.x starting`	DKIM signer active.
`DKIM=PASS` in email header	Successful signature validation.
`SASL authentication succeeded`	Relay credentials verified.

APPENDIX D – FILE AND DIRECTORY LAYOUT REFERENCE

This appendix provides a detailed overview of all relevant directories and configuration files used by the DKIM Podman environment.
It distinguishes between **host paths** and **container paths**, showing how persistent volumes are mounted and what each file does.

D.1 Host Directory Structure Overview

All persistent configuration data, keys, and logs live under `/srv/dkim`, while source files and build assets reside in `/var/Software`.

```
/
├── etc/
│   ├── containers/
│   │   └── systemd/              # Optional systemd integration for containers
│   └── cron.monthly/
│       └── dkim-logrotate.sh     # Example log rotation script
├── srv/
│   └── dkim/
│       ├── mydomain.com/
│       │   ├── postfix/          # Postfix configuration and relay credentials
│       │   ├── opendkim/         # DKIM configuration and keys
│       │   └── logs/             # Persistent logs for this domain
│       ├── mydomain.com/
│       │   ├── postfix/
│       │   ├── opendkim/
│       │   └── logs/
│       ├── ezeasproducts.com.au/
│       │   ├── postfix/
│       │   ├── opendkim/
│       │   └── logs/
│       └── example.org/
│           ├── postfix/
│           ├── opendkim/
│           └── logs/
├── var/
│   └── Software/
```

```
|       └── podman_dkim_bundle_leap15_6/
|           ├── Containerfile         # Build definition for the DKIM relay image
|           ├── docker-entrypoint.sh  # Launches OpenDKIM + Postfix
|           ├── postinstall-dkim.sh   # Automates setup and build
|           ├── README.md             # Reference for manual steps
|           └── testing-guide.txt     # Testing summary
└── backup/
    └── dkim-latest.tar.gz           # Recommended backup file location
```

D.2 Host Volume Mapping Table

Each domain's container mounts its respective configuration folders from the host to keep state persistent across rebuilds or reboots.

Host Path	Container Path	Description
`/srv/dkim/<domain>/postfix`	`/etc/postfix`	Postfix configuration (main.cf, master.cf, sasl_passwd)
`/srv/dkim/<domain>/opendkim`	`/etc/opendkim`	OpenDKIM configuration and keys
`/srv/dkim/<domain>/logs`	`/var/log`	Persistent logging for Postfix and OpenDKIM

D.3 Container Filesystem Layout

Once a container is running, the directory tree inside looks like this:

```
/
├── etc/
|   ├── postfix/
|   |   ├── main.cf
|   |   ├── master.cf
|   |   ├── sasl_passwd
|   |   ├── sasl_passwd.db
|   |   └── dynamic maps, spool configs
|   ├── opendkim/
|   |   ├── opendkim.conf
|   |   ├── KeyTable
|   |   ├── SigningTable
|   |   ├── TrustedHosts
|   |   └── keys/
|   |       └── <domain>/<selector>.private
```

```
│   └── ssl/
│       └── certs/ca-certificates.crt
├── run/
│   └── opendkim/                 # Runtime socket directory (tmpfs)
├── usr/sbin/
│   ├── postfix                  # Main Postfix executable
│   ├── opendkim                 # DKIM filter binary
│   └── postmap, postconf, etc.
└── var/log/
    ├── maillog
    ├── mail.err
    ├── mail.warn
    └── opendkim.log
```

D.4 Key File Paths

Each domain's signing keys live under `/srv/dkim/<domain>/opendkim/keys/<domain>/`. You'll typically find:

File	Description
`<selector>.private`	The private RSA signing key used by OpenDKIM
`<selector>.txt`	Public key to publish as DNS TXT record
`<selector>.pem`	(Optional) PEM version if needed for import elsewhere

Example:

```
/srv/dkim/mydomain.com/opendkim/keys/mydomain.com/
├── eacb2025.private
└── eacb2025.txt
```

D.5 Configuration File Summary

Postfix Files

File	Description
`/etc/postfix/main.cf`	Core configuration for mail relay, SASL, and DKIM
`/etc/postfix/master.cf`	Defines active services and daemons
`/etc/postfix/sasl_passwd`	Credentials for NO-IP relay authentication
`/etc/postfix/sasl_passwd.db`	Compiled version used by Postfix

OpenDKIM Files

File	Description
`/etc/opendkim/opendkim.conf`	Main configuration file

File	Description
`/etc/opendkim/KeyTable`	Maps selectors to private keys
`/etc/opendkim/SigningTable`	Maps senders to selectors
`/etc/opendkim/TrustedHosts`	Defines trusted IP ranges and clients

D.6 Log File Reference

Logs for each container are written both to:

1. The Podman-managed log stream (viewable with `podman logs dkim-<domain>`)
2. The persistent log directory mounted at `/srv/dkim/<domain>/logs`.

Typical logs:

File	Description
`/srv/dkim/<domain>/logs/maillog`	Combined mail log (Postfix + DKIM)
`/srv/dkim/<domain>/logs/mail.err`	Errors only
`/srv/dkim/<domain>/logs/opendkim.log`	Detailed DKIM signing output

D.7 Backup Contents

When creating backups with:

```
tar czf /backup/dkim-$(date +%F).tar.gz /srv/dkim
/var/Software/podman_dkim_bundle_leap15_6
```

the archive includes:

- All DKIM keys and configs
- Postfix credentials
- Podman build files and scripts
- Testing references and bundle docs

To verify contents:

```
tar -tzf /backup/dkim-YYYY-MM-DD.tar.gz | less
```

D.8 Disk Space Planning

Directory	Typical Size	Notes
`/srv/dkim/<domain>/opendkim`	100 KB–1 MB	Depends on number of keys
`/srv/dkim/<domain>/postfix`	1–2 MB	Config and queue files
`/srv/dkim/<domain>/logs`	5–20 MB per week	Rotate monthly
`/var/Software/podman_dkim_bundle_leap15_6`	~650 MB	Includes container build image
`/backup`	100 MB+	Depends on number of domains

Directory	Typical Size	Notes
		and logs

D.9 Quick File Location Lookup

Purpose	File/Path
Postfix config	`/srv/dkim/<domain>/postfix/main.cf`
Relay credentials	`/srv/dkim/<domain>/postfix/sasl_passwd`
DKIM configuration	`/srv/dkim/<domain>/opendkim/opendkim.conf`
DKIM keys	`/srv/dkim/<domain>/opendkim/keys/<domain>/`
Log files	`/srv/dkim/<domain>/logs/`
Containerfile	`/var/Software/podman_dkim_bundle_leap15_6/Containerfile`
Entrypoint script	`/var/Software/podman_dkim_bundle_leap15_6/docker-entrypoint.sh`
Build script	`/var/Software/podman_dkim_bundle_leap15_6/postinstall-dkim.sh`
Network file	`dkimnet` (Podman macvlan configuration)

APPENDIX E – REFERENCE MATERIALS AND RESOURCES

This appendix contains technical references, authoritative documentation, and upgrade guidance for the DKIM Podman relay system.
It is designed to support both current maintenance and future platform upgrades.

E.1 Authoritative Documentation Sources

Component	Reference URL	Notes
Postfix	https://www.postfix.org/documentation.html	Comprehensive parameter reference.
OpenDKIM	http://www.opendkim.org/docs.html	Upstream documentation and FAQ.
Podman	https://docs.podman.io/en/latest/	Container management, networking, and systemd integration.
openSUSE Leap 15.6	https://doc.opensuse.org/	Official admin and deployment guides.
NO-IP SMTP Relay	https://www.noip.com/support	SMTP authentication and SSL/TLS relay details.
RFC 6376 (DKIM)	https://datatracker.ietf.org/doc/html/rfc6376	The official DKIM standard definition.

E.2 Recommended Diagnostic Utilities

Install these optional packages to enhance monitoring and testing:

Tool	Command	Purpose
swaks	`zypper install swaks`	Scripted mail testing (SMTP/DKIM/SSL).
dig	`zypper install bind-utils`	DNS lookup and verification.
openssl	(preinstalled)	Test TLS/SSL connectivity to relay.
mlocate / locate	`zypper install mlocate`	Quickly find config files or logs.
logrotate	`zypper install logrotate`	Automate log management.
ncdu	`zypper install ncdu`	Disk usage visualisation under `/srv`.

E.3 Common Log Files

File	Description
`/srv/dkim/<domain>/logs/maillog`	Primary mail log (Postfix + OpenDKIM).
`/srv/dkim/<domain>/logs/opendkim.log`	Dedicated DKIM signing activity log.
`/srv/dkim/<domain>/logs/mail.err`	Error messages only.
`/var/log/journal/`	Host-level journald logs (persistent).

To monitor all DKIM containers simultaneously:

```
podman logs --tail=50 --names
```

E.4 Future Migration and Upgrades

The environment has been tested on **openSUSE Leap 15.6**.
If upgrading to Leap 16 or SLES 15 SP6, note the following:

Area	Expected Change	Action
Podman Backend	Uses *netavark* and *aardvark-dns* exclusively.	Compatible — no change required.
Quadlet	Available as standard package in Leap 16+.	Replace manual systemd units with `.container` units if preferred.
OpenDKIM Package Location	Likely renamed to `opendkim-tools`.	Verify package names and rebuild image accordingly.
Network Permissions	`macvlan` network creation may require `CAP_NET_ADMIN`.	Run `sudo` for `podman network create`.
SELinux / AppArmor	Possible stricter defaults in Leap 16.	Add `:Z` flag to all volume mounts (already done).

Before migration:

1. Backup `/srv/dkim` and `/var/Software`.
2. Export all Podman containers (`podman export` if needed).
3. Rebuild the DKIM image on the new platform from `Containerfile`.
4. Restore domain configs.
5. Run live tests using `swaks`.

E.5 Security Recommendations

- Limit SSH access to DKIM hosts (use keys, not passwords).
- Keep DKIM private keys (`.private`) restricted to `root:root` with `chmod 600`.
- Use strong NO-IP relay passwords and rotate yearly.
- Implement host-level firewall allowing outbound **port 465 only**.
- Store all DKIM backups in encrypted form using `gpg` or `veracrypt`.
- Consider monitoring DKIM integrity via external services such as:
    - https://dkimvalidator.com/
    - https://mxtoolbox.com/dkim.aspx

E.6 Document Maintenance

When updating this manual, ensure version tracking:

Version	Date	Summary
1.0	Oct 2025	Initial deployment (Leap 15.6, multi-domain Podman DKIM relays).

Version	Date	Summary
1.1	TBD	Migration update for Leap 16 (pending).

Keep the document under version control (e.g. Git or Fossil) alongside your `/var/Software` directory to maintain configuration consistency.

E.7 Useful Command Snippets

Purpose	Command
Inspect all containers	`podman ps --format "{{.Names}} {{.Status}}"`
Test DKIM DNS record	`dig +short TXT <selector>._domainkey.<domain>`
Quick restart all relays	`for c in $(podman ps -q); do podman restart $c; done`
Verify DKIM signing process	`grep "DKIM-Signature" /srv/dkim/*/logs/maillog`
Check relay authentication	`grep "SASL authentication succeeded" /srv/dkim/*/logs/maillog`

E.8 Additional Learning Resources

- *"Postfix: The Definitive Guide"* by Kyle D. Dent (O'Reilly)
- *"Email Security with DKIM, SPF and DMARC"* by J. Klensin, RFC 6376 and RFC 7489
- *openSUSE Leap Admin Guide*: https://doc.opensuse.org/documentation/leap/
- *Podman Desktop and Quadlet Tutorials*: https://podman.io/tutorials/

E.9 Support and Escalation Paths

Type	Contact / Resource	Description
Internal (Eric / c4israel IT)	`ebvl-dkimhost-001.cdn.mydomain.com`	Primary DKIM management host.
Secondary (C4I Host)	`xxxx-dkimhost-001`	Secondary relay for mydomain.com.
NO-IP Support	https://www.noip.com/support	SMTP relay credentials and relay issues.
openSUSE Forums	https://forums.opensuse.org/	OS and package-level issues.
Podman Community	https://github.com/containers/podman/discussions	Container-level support.

Appendix F - Network Setup - Routing Changes

Excellent catch — and yes, you're absolutely right to stop and clarify this before going live.
Let's unpack the issue properly and then map out a clean solution.

Why This Matters

When using **Podman macvlan** networks, each container behaves as if it's directly connected to the physical NIC with its own unique MAC and IP address.
However, there's a key restriction:

> **The host and its macvlan containers cannot directly communicate if they're on the same subnet.**

This means:

- Your host (`xxx.xxx.xxx.27`) and containers (`xxx.xxx.xxx.x`) **cannot see each other** directly.
- Even though they share the same physical NIC, Linux isolates macvlan interfaces at Layer 2.
- You'll get unreachable errors when the host tries to ping a container, or vice versa.

So — you're absolutely right: putting the DKIM container IPs inside the same subnet as the host (`xxx.xxx.xxx.0/24`) is not appropriate.

Correct Network Design

You should instead dedicate a *secondary subnet* (e.g. `xxx.xxx.xxx.0/24`) to the Podman macvlan network.
Here's how this layout would look:

Role	Interface	Subnet	Example IP
Host (openSUSE)	`eth0`	xxx.xxx.xxx.0/24	xxx.xxx.xxx.27
Default Gateway	—	xxx.xxx.xxx.254	xxx.xxx.xxx.254
DKIM Network	macvlan via eth0	xxx.xxx.xxx.0/24	xxx.xxx.xxx.xxx–.117
DKIM Container 1 (mydomain.com)	macvlan	xxx.xxx.xxx.xxx	—
DKIM Container 2 (mydomain.com)	macvlan	xxx.xxx.xxx.xxx	—
DKIM Container 3 (ezeasproducts.com.au)	macvlan	xxx.xxx.xxx.117	—

This achieves **isolation with routability**, since:

- The containers will use the host's NIC as their physical bridge.
- The host will route between xxx.xxx.xxx.0 and xxx.xxx.xxx.0 internally.

- Your main gateway (`xxx.xxx.xxx.254`) still provides Internet access.

How to Implement It

1. Create the new macvlan network

```
sudo podman network create \
  --driver macvlan \
  --subnet xxx.xxx.xxx.0/24 \
  --gateway xxx.xxx.xxx.1 \
  -o parent=eth0 dkimnet
```

(Use `.1` as a dummy gateway address; the real routing goes via the host.)

---

2. Add a route on the host to reach the new subnet

```
sudo ip route add xxx.xxx.xxx.0/24 dev eth0
```

This tells the host kernel that containers are reachable via the same physical NIC.

---

3. Enable inter-subnet routing if required

If other LAN systems need to reach the DKIM containers directly (for example, if GroupWise runs on another physical host), you have two options:

- **Option A: Host-based NAT (simpler):**
  Containers can reach the Internet, but other LAN systems won't reach them directly.
  This is fine since DKIM relays send mail outbound only.
- **Option B: Add static route on your router:**

  ```
  Destination: xxx.xxx.xxx.0/24
  ```

  ```
  Next Hop: xxx.xxx.xxx.27
  ```

  This allows other LAN systems to communicate directly with containers if needed (e.g., GroupWise servers on other hosts).

---

4. Use container IPs in the new subnet

When you create the containers, update the `--ip` assignments like this:

Container	IP
dkim-mydomain-com	xxx.xxx.xxx.xxx
dkim-mydomain-com	xxx.xxx.xxx.xxx
dkim-ezeasproducts-com-au	xxx.xxx.xxx.11

Container	IP
	7
example.org (optional)	xxx.xxx.xxx.X

---

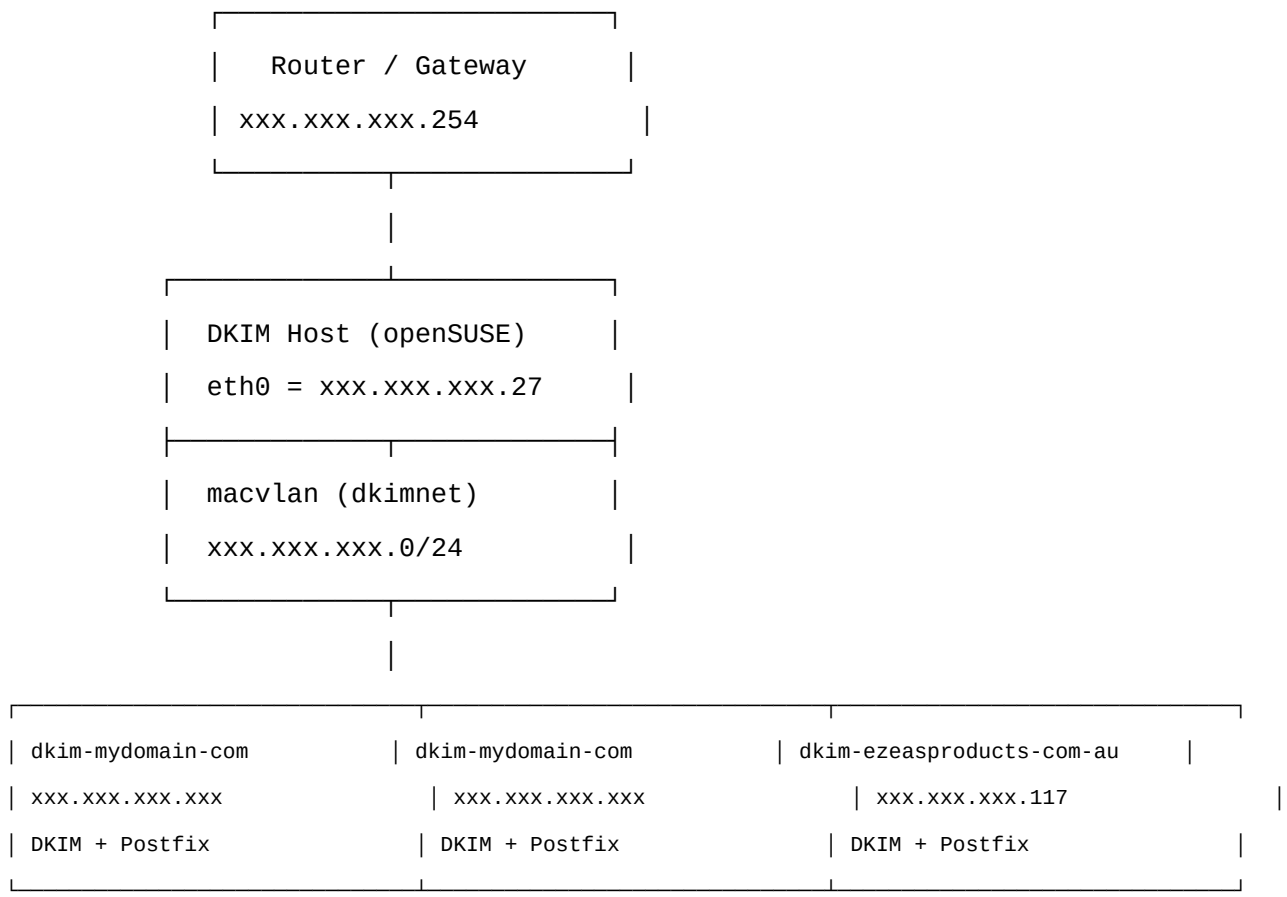
5. Update GroupWise GWIA relay configuration

GroupWise now points to the **host IP (xxx.xxx.xxx.27)** for the relay, *not* the containers.
The host will transparently route traffic to the appropriate DKIM container.

If GroupWise is running on the same host, you may need to add a loopback route:

```
sudo ip route add xxx.xxx.xxx.0/24 dev eth0
```

✅ Resulting Topology



✅ Summary of Actions

Action	Command / Change
Create separate subnet	xxx.xxx.xxx.0/24
Update Podman network	`podman network create --subnet xxx.xxx.xxx.0/24 ...`
Update container IPs	xxx.xxx.xxx.xxx-117
Add host route	`ip route add xxx.xxx.xxx.0/24 dev eth0`

Action	Command / Change
Optional router route	`xxx.xxx.xxx.0/24 → xxx.xxx.xxx.27`
Update GWIA relay	Use host IP (xxx.xxx.xxx.27) as outbound relay

APPENDIX G - Implement a normal MTA (Mail Transfer Agent).

Below is exactly what you need to modify, plus some advice on DNS, SPF, and firewalls when sending directly.

🧭 Goal

> DKIM container should deliver mail directly to recipients' MX servers — *no upstream relay/smart host.*

1️⃣ Edit the Postfix Configuration

You'll change `/etc/postfix/main.cf` inside the container.

Step 1. Remove or comment out relayhost and SASL

```
# relayhost = [smtp-auth.no-ip.com]:465
# smtp_sasl_auth_enable = yes
# smtp_sasl_password_maps = hash:/etc/postfix/sasl_passwd
# smtp_sasl_security_options = noanonymous
# smtp_tls_CAfile = /etc/ssl/certs/ca-certificates.crt
```

Step 2. Add or verify direct delivery settings

Keep these general lines:

```
inet_interfaces = all
inet_protocols = ipv4
mynetworks = 127.0.0.0/8, xxx.xxx.xxx.0/24, 10.0.0.0/8

smtp_tls_security_level = may
smtp_tls_loglevel = 1
smtp_tls_note_starttls_offer = yes
smtp_tls_session_cache_database = btree:${data_directory}/smtp_scache

relay_domains = $mydomain
default_destination_concurrency_limit = 10
default_destination_rate_delay = 1s
compatibility_level = 3.6
```

That tells Postfix to:

- Discover destination MX records using DNS.

- Attempt opportunistic STARTTLS.
- Send directly to those servers.

2️⃣ (Optional) Enable DNS Resolver Inside Container

Ensure `/etc/resolv.conf` inside the container has reachable nameservers:

```
nameserver xxx.xxx.xxx.252
nameserver xxx.xxx.xxx.253
nameserver 8.8.8.8
```

If your containers use macvlan, verify they have internet routing:

```
ping 8.8.8.8
dig gmail.com MX
```

3️⃣ (Optional) Adjust Outbound DNS, PTR, SPF, DMARC

Because you'll now be **sending mail directly from your own IP**, receiving systems will check:

- ✅ **Reverse DNS (PTR)** must resolve to your public hostname (e.g. `dkim.mydomain.com`).
- ✅ **SPF** must include your public IPs:

  ```
  v=spf1 ip4:x.x.x.x ip4:y.y.y.y include:noip-smtp include:_spf.google.com -all
  ```

- ✅ **DMARC** should match your domain alignment.

If NO-IP handled those before, you'll now need to ensure your ISP or hosting provider delegates PTR records to you or updates them on request.

4️⃣ (Optional but Recommended) Enable Authentication for GroupWise / MassMail

If your internal systems will feed these DKIM relays directly, you may want them to authenticate or be whitelisted.

Inside `main.cf`:

```
smtpd_recipient_restrictions =
    permit_mynetworks,
    reject_unauth_destination
```

That already permits trusted LAN senders (`mynetworks`), so no login is required from your internal hosts.

5️⃣ Restart Postfix

```
sudo podman exec -it dkim-niyshah-com-au bash -lc 'postfix reload'
```

6️⃣ Verify Outbound Delivery

Run a live test:

```
swaks --from test@mydomain.com --to yourgmail@gmail.com --server <container-ip> --tls
```

You should see in `/var/log/mail`:

```
postfix/smtp[...] Trusted TLS connection established to gmail-smtp-
in.l.google.com[142.250.x.x]:25
```

And in Gmail “Show original”:
✅ `dkim=pass`
✅ `spf=pass` (if your SPF is updated)
✅ `dmarc=pass`

7️⃣ If You Later Revert to NO-IP Relay

Simply uncomment:

```
relayhost = [smtp-auth.no-ip.com]:465
```

```
smtp_sasl_auth_enable = yes
```

```
smtp_sasl_password_maps = hash:/etc/postfix/sasl_passwd
```

```
smtp_sasl_security_options = noanonymous
```

```
smtp_tls_security_level = encrypt
```

…and restart Postfix again.

Quick Checklist for Direct-to-Internet Delivery

Item	Required	Notes
Outbound port 25 open	✅	Confirm with `telnet gmail-smtp-in.l.google.com 25`
Reverse DNS (PTR) set	✅	Must match the hostname used in `myhostname`
SPF includes public IP	✅	Receiving servers will check this
DKIM DNS record valid	✅	Already configured correctly
DMARC record present	✅	e.g. `v=DMARC1; p=quarantine; rua=mailto:dmarc@mydomain.com`

APPENDIX H - Additional Notes

```
sudo podman run -d --name dkim-mydomain-com \
  --network dkimnet --ip xxx.xxx.xxx.xxx \
  --tmpfs /run/opendkim:mode=0755 \
  -v /srv/dkim/mydomain.com/opendkim:/etc/opendkim:Z \
  -v /srv/dkim/mydomain.com/postfix:/etc/postfix:Z \
  -v /srv/dkim/mydomain.com/logs:/var/log:Z \
  -e TZ=Australia/Brisbane \
localhost/postfix-opendkim:leap15_6

sudo podman logs --tail=50 dkim-mydomain-com
sudo chmod 600 /srv/dkim/mydomain.com/postfix/sasl_passwd
sudo podman exec -it dkim-mydomain-com postmap /etc/postfix/sasl_passwd
sudo podman exec -it dkim-mydomain-com postfix reload

sudo mkdir -p /srv/dkim/mydomain.com/opendkim/keys/mydomain.com
sudo podman run --rm \
  --entrypoint opendkim-genkey \
  -v /srv/dkim/mydomain.com/opendkim:/work:Z \
  localhost/postfix-opendkim:leap15_6 \
  -b 2048 -s xxxx2025 -d mydomain.com -D /work/keys/mydomain.com

sudo chmod 600 /srv/dkim/mydomain.com/opendkim/keys/mydomain.com/xxxx2025.private

podman logs --tail=80 dkim-mydomain-com

podman exec -it dkim-mydomain-com bash
chown root:postdrop /usr/sbin/postqueue
chown root:postdrop /usr/sbin/postdrop
chmod g+s /usr/sbin/postqueue
```

```
chmod g+s /usr/sbin/postdrop
exit

podman restart dkim-mydomain-com
```

Enabling Inbound TLS

1. **Add STARTTLS to main.cf**

Add these lines to your `/srv/dkim/mydomain.com/postfix/main.cf`:

```bash
sudo tee -a /srv/dkim/mydomain.com/postfix/main.cf > /dev/null <<'EOF'

# --- Inbound STARTTLS (for clients like swaks) ---
smtpd_tls_security_level = may
smtpd_tls_cert_file = /etc/postfix/smtpd.crt
smtpd_tls_key_file = /etc/postfix/smtpd.key
smtpd_tls_loglevel = 1
smtpd_tls_session_cache_database = lmdb:${data_directory}/smtpd_scache
EOF
```

2. **Enable STARTTLS in master.cf**

Edit `/srv/dkim/mydomain.com/postfix/master.cf` and make sure the smtp line has `smtp inet` with `smtpd` and `-o smtpd_tls_security_level=may`:

```bash
sudo podman exec -it dkim-mydomain-com bash -c '
cat > /etc/postfix/master.cf << EOF
smtp      inet  n       -       n       -       -       smtpd
  -o smtpd_tls_security_level=may
EOF
'
```

3. **Generate Self-Signed Certificate (if missing)**

# Generate certificate in the postfix directory
sudo openssl req -x509 -nodes -days 3650 -newkey rsa:2048 \
    -keyout /srv/dkim/mydomain.com/postfix/smtpd.key \
    -out /srv/dkim/mydomain.com/postfix/smtpd.crt \
    -subj "/CN=dkim.mydomain.com" \
    -addext "subjectAltName = DNS:dkim.mydomain.com, IP:xxx.xxx.xxx.xxx"

# Set permissions
sudo chmod 644 /srv/dkim/mydomain.com/postfix/smtpd.crt
sudo chmod 600 /srv/dkim/mydomain.com/postfix/smtpd.key
‘

4. **Restart Container**

```
podman restart dkim-mydomain-com
```

5. **Test STARTTLS**

```bash
swaks --from test@mydomain.com --to ebelcher@mydomain.com --server xxx.xxx.xxx.xxx -tls
```

Alternative: Use TLS Port (465) for Testing

Since you're already using TLS for outbound, test with the TLS port:

```bash
swaks --from test@mydomain.com --to ebelcher@mydomain.com --server xxx.xxx.xxx.xxx --port 465 --tls
```

Check Current TLS Configuration

```bash
sudo podman exec -it dkim-mydomain-com postconf | grep tls
```

Test with openssl (verify TLS)

```bash
# Test STARTTLS
openssl s_client -connect xxx.xxx.xxx.xxx:25 -starttls smtp

# Test TLS directly on 465
openssl s_client -connect xxx.xxx.xxx.xxx:465
```