



Postfix 2025

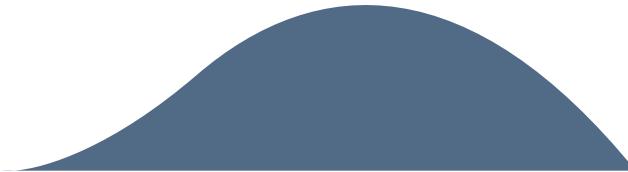
✉ Une formation présentée par Ascent et Andromed.

Appuyez sur espace pour la page suivante →

Jimmylan Surquin

Fondateur  [Andromed](#)

- Lille, France 
- Création de contenu sur  [jimmylansrq](#)
- Blog & Portfolio [jimmylan.fr](#)



DISCLAIMER

Dans cette formation nous allons voir les concepts fondamentaux et avancés de Postfix en 2025.

Postfix est un serveur de messagerie (MTA - Mail Transfer Agent) robuste, sécurisé et performant, créé par Wietse Venema.



FORMATION POSTFIX INTENSIVE - 2 JOURS (14H)

 Introduction à Postfix (30min)

 Installation et config de base (1h30)

 Architecture essentielle (45min)

 Configuration main.cf (2h)

 Alias et domaines virtuels (1h)

 TLS et sécurité (1h)

 DKIM, SPF, DMARC (1h30)

 Dovecot (1h30)

 Anti-spam essentiel (1h)

 Logs et monitoring (45min)

 Sauvegarde (30min)

 TP pratiques (2h)

 Troubleshooting (1h)

 QCM final (30min)

PROGRAMME FORMATION INTENSIVE

JUL
17

JOUR 1 (7h) - FONDAMENTAUX & SÉCURITÉ

- **9h00-9h30** : Introduction Postfix (30min)
- **9h30-11h00** : Installation & config de base (1h30)
- **11h15-12h00** : Architecture essentielle (45min)
- **13h00-15h00** : Configuration main.cf (2h)
- **15h15-16h15** : Alias et domaines virtuels (1h)
- **16h30-17h30** : TLS et sécurité (1h)
- **17h45-19h15** : DKIM, SPF, DMARC (1h30)

JOUR 2 (7h) - PRATIQUE & PRODUCTION

- **9h00-10h00** : Dovecot (1h)
- **10h15-11h15** : Anti-spam essentiel (1h)
- **11h30-12h00** : Logs et monitoring (30min)
- **12h00-13h00** : Sauvegarde (1h)
- **13h15-14h15** : TP pratiques (1h)
- **14h30-15h00** : Troubleshooting (30min)
- **15h15-15h45** : QCM final (30min)

🎯 OBJECTIFS DE LA FORMATION INTENSIVE

À la fin de ces 2 jours, vous saurez :

- ✓ Installer et configurer Postfix sur Linux
- ✓ Comprendre l'architecture modulaire de Postfix
- ✓ Configurer le main.cf pour vos besoins
- ✓ Gérer les domaines virtuels et alias
- ✓ Sécuriser avec DKIM, SPF, DMARC et TLS
- ✓ Configurer Dovecot pour la gestion des boîtes mail
- ✓ Protéger contre le spam efficacement
- ✓ Surveiller et diagnostiquer les problèmes
- ✓ Sauvegarder et restaurer votre configuration





RESSOURCES COMPLÉMENTAIRES

Documentation officielle :

- [Postfix Documentation](#)
- [Postfix Configuration Parameters](#)

Outils utiles :

- [MXToolbox](#) - Tests DNS et blacklists
- [Mail-tester](#) - Test de qualité des emails
- [Email-tester](#) - Test de qualité des emails via une adresse simple : check-auth@verifier.port25.com
- [DKIM Validator](#) - Validation DKIM

Introduction à Postfix

 Découverte du serveur de messagerie le plus utilisé au monde

Qu'est-ce que Postfix ?

Postfix est un **MTA** (Mail Transfer Agent) créé par **Wietse Venema** dans les années 1990.

Un MTA, c'est quoi exactement ?

Imaginez le système postal : quand vous envoyez une lettre, elle passe par plusieurs centres de tri avant d'arriver à destination.

Un MTA fait exactement la même chose, mais pour vos emails !

Qu'est ce qu'un DNS ?

1 DNS = Domain Name System

- Littéralement : système de noms de domaine
- C'est comme un annuaire téléphonique d'internet : il traduit des noms faciles à retenir (ex : example.fr) en adresses IP (ex : 93.22.332.10) que les ordinateurs comprennent.



2 Comment ça fonctionne concrètement

- Quand vous tapez example.fr dans votre navigateur ou que vous envoyez un mail à user@example.fr ou que vous faites une requête HTTP à http://example.fr :

1. Votre ordinateur demande au DNS : "quelle est l'IP de example.fr ?"
2. Le DNS répond : "c'est 93.22.332.10" (via un enregistrement A).
3. Ensuite, votre ordinateur contacte directement cette IP.

- Pour les mails :
- DNS contient aussi des enregistrements MX pour dire quel serveur gère les mails de ce domaine.
- Exemple : example.fr → MX → mail.example.fr → A → 93.22.332.10

3 En résumé simple

- DNS = annuaire
- Nom de domaine = nom que vous tapez (ex : example.fr)
- A record = IP réelle du serveur (ex : 93.22.332.10)
- MX record = serveur de mail pour ce domaine (ex : mail.example.fr)

Les 3 acteurs principaux du mail

- **MTA** (Mail Transfer Agent) : Le facteur qui achemine le courrier
 - Exemple : Postfix, Sendmail, Exim
- **MUA** (Mail User Agent) : Votre boîte aux lettres personnelle
 - Exemple : Thunderbird, Outlook, Gmail (interface web)
- **MDA** (Mail Delivery Agent) : Le préposé qui dépose le courrier dans votre boîte
 - Exemple : Dovecot, Procmail



Postfix : Une histoire de sécurité

Contexte historique

Dans les années 1990, **Sendmail** dominait le marché mais souffrait de nombreux problèmes de sécurité.

Wietse Venema, chercheur en sécurité chez IBM, a décidé de créer une alternative :

- Plus **sécurisée** par conception
- Plus **rapide**
- Plus **simple** à configurer

🎯 La philosophie de Postfix

Postfix est conçu selon des principes stricts :

Sécurité avant tout

- Séparation des privilèges (chaque processus a un rôle unique)
- Privilèges minimum (chaque processus n'a que les droits nécessaires)
- Architecture modulaire (isolation des composants)

Performance

- Traitement asynchrone des messages
- Files d'attente optimisées
- Capable de gérer des millions d'emails par jour

Simplicité

- Configuration claire et lisible
- Moins de 100 fichiers de configuration (contre des centaines pour Sendmail)
- Compatibilité avec Sendmail pour faciliter la migration



Pourquoi choisir Postfix en 2025 ?

✓ Les avantages

Robustesse éprouvée

- Utilisé par des millions de serveurs dans le monde
- Gmail de Google l'utilise (avec des modifications)
- Stabilité légendaire (uptime de plusieurs années possible)

Sécurité exceptionnelle

- Très peu de failles de sécurité dans son histoire
- Architecture conçue pour limiter les impacts en cas de problème
- Mises à jour régulières et communauté active

Performance

- Gère facilement des milliers d'emails par minute
- Consommation mémoire raisonnable
- Scalabilité horizontale possible



Flexibilité

- Configuration très granulaire
- Supporte tous les standards modernes (DKIM, SPF, DMARC, TLS 1.3)
- Intégration facile avec d'autres outils (antivirus, antispam, bases de données)

Les inconvénients

Courbe d'apprentissage : Configuration peut sembler complexe

- Beaucoup de paramètres à comprendre
- Nécessite une bonne compréhension du fonctionnement des emails

Documentation dense : Documentation officielle très complète mais parfois technique - Nécessite du temps pour maîtriser tous les aspects

Postfix vs les alternatives en 2025

Postfix vs Sendmail : Postfix gagne sur tous les critères (Sécurité ★★★★★ vs ★★★★, Performance ★★★★★ vs ★★★★, Simplicité ★★★★★ vs ★★★)

Postfix vs Exim : Postfix plus simple, Exim plus flexible mais plus complexe. Postfix suffisant pour 95% des cas d'usage.

Postfix vs solutions cloud : Choisir Postfix pour le contrôle total, la confidentialité, les gros volumes et les compétences techniques. Choisir le cloud pour la simplicité, les petits volumes (< 10 000/mois) et déléguer la gestion.

Les cas d'usage de Postfix

 **Entreprise** : Serveur mail interne (comptes utilisateurs, Active Directory/LDAP, conformité)

-  **Hébergeur web** : Service mail multi-clients (domaines virtuels, quotas, isolation)
-  **Application web** : Emails transactionnels (notifications, confirmations, newsletters)
-  **Infrastructure sécurisée** : Confidentialité maximale (médical HIPAA, financier PCI-DSS, gouvernement)

L'écosystème Postfix en 2025

Postfix ne fonctionne généralement pas seul : **Postfixadmin** (interface web)

- **Roundcube/Rainloop** (webmail)
- **Grafana+Prometheus** (monitoring)
- **Rspamd** (anti-spam basique)

Architecture typique en 2025

Une stack email moderne ressemble à ça :

```
Internet
  ↓
[Firewall]
  ↓
[Postfix MTA] ↔ [Fichiers de configuration locaux]
  ↓
[Protection anti-spam basique]
  ↓
[Stockage emails local]
  ↓
[Clients : Thunderbird, Outlook, Webmail]
```



Les standards à connaître

Protocoles essentiels

SMTP (Simple Mail Transfer Protocol)

- Port 25 : Communication entre serveurs MTA
- Port 587 : Soumission par les clients (SMTP submission)
- Port 465 : SMTP sécurisé (deprecated mais encore utilisé)

IMAP / POP3

- IMAP (port 143/993) : Synchronisation des emails sur plusieurs appareils

Par exemple : Thunderbird, Outlook, Gmail (interface web)

- POP3 (port 110/995) : Téléchargement et suppression des emails du serveur

Par exemple : Roundcube, Rainloop, Horde, Squirrelmail, etc.



Donc quand choisir IMAP ou POP3 ?

- IMAP : Pour les utilisateurs qui ont besoin de synchroniser leurs emails sur plusieurs appareils.
- POP3 : Pour les utilisateurs qui ont besoin de télécharger leurs emails sur un seul appareil.

La plus part du temps, on utilise IMAP.



Sécurité des emails

SPF (Sender Policy Framework) : Enregistrement DNS indiquant quels serveurs peuvent envoyer des emails pour votre domaine

DKIM (DomainKeys Identified Mail) : Signature cryptographique des emails pour prouver leur authenticité

DMARC (Domain-based Message Authentication, Reporting & Conformance) : Politique de validation combinant SPF et DKIM

TLS (Transport Layer Security) : Chiffrement des communications entre serveurs, TLS 1.3 est le standard en 2025



Statistiques et adoption

Postfix dans le monde

- **Part de marché** : ~35% des serveurs mail sur Internet
- **Performances** : Capable de gérer 1M+ emails/jour sur un serveur modeste
- **Fiabilité** : Taux d'uptime moyen > 99.9%

Qui utilise Postfix ?

- Grandes entreprises (Fortune 500)
- Hébergeurs web (OVH, 1&1, etc.)
- Universités et institutions éducatives
- Gouvernements et organismes publics
- Startups et PME du monde entier

Les compétences nécessaires

Pour administrer Postfix efficacement, vous devez maîtriser :

Linux / Unix : Lignes de commande, gestion des services (systemd), permissions et utilisateurs

Réseaux : DNS (enregistrements MX, A, TXT), protocoles TCP/IP, firewall et routage

Sécurité : Certificats SSL/TLS, authentification et autorisation, bonnes pratiques de hardening

Fichiers de configuration : Configuration via fichiers texte simples et tables de correspondance

Prérequis pour cette formation

Connaissances

- Bases de Linux (navigation, édition de fichiers)
- Notions de réseaux (IP, ports, DNS)
- Comprendre ce qu'est un email et comment il fonctionne

Environnement

Pour suivre cette formation, vous aurez besoin de :

- Une machine Linux (Ubuntu 22.04+ / Debian 12+ / Rocky Linux 9+)
- Accès root ou sudo
- Connexion Internet
- 2 GB RAM minimum, 4 GB recommandé
- 20 GB d'espace disque

Nous utiliserons **Docker** pour certains exercices afin de faciliter les tests et les démonstrations.

Objectifs de la formation initiation

À la fin de la formation initiation, vous serez capable de :

- Installer et configurer Postfix sur Linux
- Comprendre l'architecture et le fonctionnement interne
- Configurer les domaines virtuels et les alias
- Mettre en place la sécurité de base (TLS, authentification)
- Implémenter SPF, DKIM et DMARC
- Protéger votre serveur contre le spam
- Analyser les logs et résoudre les problèmes courants
- Sauvegarder et restaurer votre configuration

Les ressources utiles

Documentation officielle

- Site officiel : <https://www.postfix.org/>
- Documentation : <https://www.postfix.org/documentation.html>
- FAQ : <https://www.postfix.org/faq.html>

Communauté

- Mailing list officielle : postfix-users@postfix.org
- Forums : Server Fault, Unix & Linux Stack Exchange
- Reddit : r/postfix, r/selfhosted

Livres recommandés

- "The Book of Postfix" par Ralf Hildebrandt
- "Postfix: The Definitive Guide" par Kyle Dent
- "The Postfix Configuration Manual" (documentation officielle)

Ce que nous allons construire

Au cours de cette formation, nous allons construire progressivement un serveur mail complet :

Phase 1 - Les bases : Installation et configuration initiale, envoi et réception d'emails locaux, configuration des domaines

Phase 2 - La sécurité : Chiffrement TLS, authentification SASL, SPF, DKIM, DMARC

Phase 3 - La protection : Anti-spam (RBL, restrictions), anti-virus, content filtering

Phase 4 - L'intégration : Domaines virtuels avec fichiers, webmail basique

Phase 5 - La production : Monitoring basique, sauvegarde et restauration simple

Méthodologie de cette formation

🎯 Approche pédagogique

Théorie et pratique équilibrées : Chaque concept est expliqué avec des analogies, suivie d'exemples concrets, puis des exercices pratiques

Apprentissage progressif : Du plus simple au plus complexe, chaque module s'appuie sur les précédents, répétition espacée pour ancrer les connaissances

Cas réels : Situations tirées de l'expérience terrain, problèmes courants et leurs solutions, best practices de l'industrie

Format des exercices

Chaque module contient :

-  **Exercices guidés** : Pas à pas pour découvrir
-  **Exercices autonomes** : Pour pratiquer seul
-  **Challenges** : Pour aller plus loin
-  **QCM** : Pour valider vos connaissances

Conseils pour réussir

Bonnes pratiques

Pratiquez, pratiquez, pratiquez : Ne vous contentez pas de lire les slides, reproduisez tous les exemples, testez des variantes

Cassez votre serveur : N'ayez pas peur de faire des erreurs, c'est en cassant qu'on apprend, utilisez des snapshots ou Docker pour faciliter les tests

Lisez les logs : Les logs sont vos meilleurs amis, prenez l'habitude de les consulter systématiquement, apprenez à les déchiffrer

Documentez vos configurations : Commentez vos fichiers de configuration, tenez un journal de bord, notez les commandes utiles

Posez des questions : Il n'y a pas de question bête, la communauté Postfix est bienveillante, mieux vaut demander que de faire n'importe quoi

Le mot de la fin

🚀 Prêt à démarrer ?

Postfix est un outil **puissant** et **fiable**. Avec cette formation, vous allez acquérir les compétences pour :

- Gérer des serveurs mail professionnels
- Assurer la sécurité et la délivrabilité de vos emails
- Diagnostiquer et résoudre les problèmes rapidement

🎓 Certification

À la fin de chaque niveau (initiation et perfectionnement), vous passerez un QCM de validation.

Let's go !

DIRECTION LE PROCHAIN MODULE : **Installation et configuration de base**

Module suivant →



QCM - Module 1 : Introduction à Postfix

Question 1

Quel est le rôle principal d'un MTA (Mail Transfer Agent) ?

- A) Lire les emails (comme Thunderbird)
- B) Acheminer et délivrer les emails
- C) Stocker les emails pour consultation
- D) Afficher les emails dans un navigateur

Question 2

Qui a créé Postfix et pourquoi ?

- A) Linus Torvalds - Pour remplacer Exchange
- B) Wietse Venema - Pour avoir une alternative plus sécurisée à Sendmail
- C) Mark Zuckerberg - Pour Facebook
- D) Apache Foundation - Pour le serveur web



Question 3

Quel est le principe de sécurité principal de Postfix ?

- A) Tout dans un seul gros processus
- B) Séparation des priviléges et architecture modulaire
- C) Authentification par mot de passe uniquement
- D) Chiffrement obligatoire



Question 4

Quel composant remet les emails dans la boîte de réception du destinataire ?

- A) MTA
- B) MDA
- C) MUA
- D) LDA



Question 5

Quel port TCP standard Postfix utilise-t-il pour accepter les connexions SMTP non chiffrées ?

- A) 25
- B) 110
- C) 143
- D) 587



Réponses - Module 1

Question 1 : Réponse B - Le MTA (comme Postfix) **achemine et délivre** les emails entre serveurs. Le MUA lit les emails, le MDA les stocke.

Question 2 : Réponse B - **Wietse Venema** (IBM) a créé Postfix dans les années 1990 comme alternative **plus sécurisée, rapide et simple** à Sendmail.

Question 3 : Réponse B - Postfix utilise la **séparation des privilèges** : chaque processus a un rôle unique et les droits minimum nécessaires.

Question 4 : Réponse B - Le **Mail Delivery Agent (MDA)** dépose le message dans la boîte du destinataire (via un MDA local ou `dovecot-lda`). Le MTA transporte, le MUA lit.

Question 5 : Réponse A - Le **port 25/TCP** est le port SMTP historique utilisé entre MTAs. Le port 587 est dédié aux clients authentifiés (submission).

Exercice pratique - Module 1

🎯 Objectif

Identifier les composants d'une architecture email

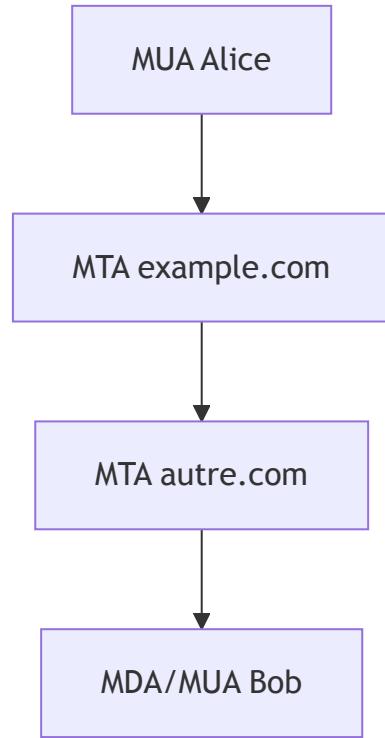
📋 Tâche

Sur papier ou tableau, dessinez le parcours d'un email de alice@example.com vers bob@autre.com en identifiant :

1. Le MUA d'Alice
2. Le MTA d@example.com (Postfix)
3. Le MTA d'autre.com
4. Le MDA/MUA de Bob

Temps : 5 minutes **Correction** : Discussion collective

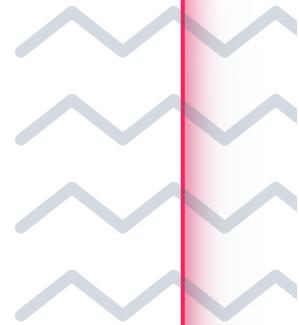
✓ Réponse :



Installation et Configuration de Base



Mettre en place votre premier serveur Postfix



Préparation de l'environnement

Avant d'installer Postfix, il faut préparer le terrain !

Vérifications préalables

Système d'exploitation

Postfix fonctionne sur tous les Unix/Linux :

- Ubuntu / Debian
- Red Hat / CentOS / Rocky Linux / AlmaLinux
- FreeBSD / OpenBSD
- macOS (pour le développement)

Pour cette formation, nous utiliserons principalement **Ubuntu 24.04 LTS** (Debian-based) et **Rocky Linux 9** (Red Hat-based).

🔍 Vérifier les prérequis système

Connexion SSH au serveur et vérification des ressources :

```
# Vérifier la version du système  
cat /etc/os-release  
  
# Vérifier les ressources disponibles  
free -h  
df -h
```

🌐 Configuration DNS préalable

Crucial ! Avant d'installer Postfix, votre DNS doit être correctement configuré.



Enregistrements DNS nécessaires

Où enregistrer ces enregistrements DNS ?

- Vous pouvez enregistrer ces enregistrements DNS chez votre hébergeur/FAI.

Pour tester localement seulement

- Vous pouvez modifier le fichier `/etc/hosts` sur votre machine pour "simuler" l'enregistrement A :

```
# Modifier le fichier /etc/hosts  
sudo nano /etc/hosts
```

```
127.0.0.1 mail.example.com
```

```
# Redémarrer le service DNS  
sudo systemctl restart systemd-resolved
```

L'enregistrement MX n'est pas nécessaire pour les tests internes.

- Le PTR n'a pas d'importance pour les tests locaux (il sert uniquement pour la réputation d'envoi vers Internet).
- Vous pourrez envoyer des mails localement à `root@localhost` ou à `user@tomaine.local` sans souci.



Nous allons utiliser un vrai domaine pour les tests, mais vous pouvez utiliser un domaine fictif comme example.com .

Dans notre cas nous allons utiliser le domaine jimmylan.fr .

Et il nous faut bien sur un VPS ou un serveur dédié pour le serveur mail.



Le domaine est enregistré chez OVH.

Donc nous allons configurer le DNS chez OVH directement , si vous avez un autre FAI, vous devrez configurer le DNS chez eux , si vous utilisez un système plus classique, vous devrez configurer le tout dans B.I.N.D. (Bind DNS)

[<https://www.it-connect.fr/dns-avec-bind-9/>](Bind9 - Configuration DNS avec Bind9)

Enregistrement A : Pointe vers l'IP de votre serveur

```
mail.jimmylan.fr. IN A 51.68.224.131
```

Enregistrement MX : Indique le serveur mail du domaine

```
jimmylan.fr. IN MX 10 mail.jimmylan.fr.
```

Le chiffre (10) est la priorité : plus c'est petit, plus c'est prioritaire.

Enregistrement PTR (Reverse DNS) : TRÈS IMPORTANT !

Le PTR fait le lien inverse : IP → nom de domaine. Sans PTR correct, vos emails seront considérés comme spam !

```
10.113.0.203.in-addr.arpa. IN PTR mail.example.com.
```

⚠ Note importante : Le PTR doit être configuré chez votre hébergeur/FAI (vous ne pouvez pas le faire vous-même).



The screenshot shows the OVH DNS management interface. The top navigation bar includes links for Tableau de bord, Bare Metal Cloud, Hosted Private Cloud, Public Cloud, Web Cloud (selected), Télécom, Sunrise, and Marketplace. The language is set to Français. A sidebar on the left lists various services under 'Commander' and 'Noms de domaine', with 'jimmylan.fr' selected. The main content area displays the 'Zone DNS' tab for the domain 'jimmylan.fr'. It shows a table of DNS records:

Domaine	TTL	Type	Cible
jimmylan.fr.	0	NS	ns11.ovh.net.
mail.jimmylan.fr.	0	CNAME	ssl0.ovh.net.
smtp.jimmylan.fr.	0	CNAME	ssl0.ovh.net.
pop3.jimmylan.fr.	0	CNAME	ssl0.ovh.net.
imap.jimmylan.fr.	0	CNAME	ssl0.ovh.net.
jimmylan.fr.	600	SPF	v=spf1 mx:atestmail.jimmylan.fr include:mx.ovh.com ~all
atlas.jimmylan.fr.	0	A	51.68.224.131
jimmylan.fr.	0	A	76.76.21.21
www.jimmylan.fr.	0	CNAME	cname.vercel-dns.com.
testmail.jimmylan.fr.	0	A	51.68.224.131

On the right side, there are buttons for 'Ajouter une entrée', 'Modifier en mode textuel', 'Modifier le TTL par défaut', 'Voir l'historique de ma zone DNS', and 'Réinitialiser ma zone DNS'. A 'Guides' section is also present.



ovh.com/manager/#/web/domain/jimmylan.fr/zone

Tableau de bord Bare Metal Cloud Hosted Private Cloud Public Cloud Web Cloud Télécom Sunrise Marketplace

Version classique Version beta Français Jimmylan Surquin

jimmylan.fr

Renouvellement automatique prévu en janv. 2026

Informations générales Zone DNS Serveurs DNS Redirection DynHost GLUE DS Records

Ajouter une entrée Modifier en mode texte Modifier le TTL par défaut Voir l'historique de ma zone DNS Réinitialiser ma zone DNS

Domaine TTL Type Cible

jimmylan.fr.	0	MX	1 testmail.jimmylan.fr.
_dmarc.jimmylan.fr.	0	DMARC	v=DMARC1; p=none; sp=none; aspf=r; v=DKIM; h=sha256;p=MILBIJANBgkhkG9w0BAAQFfAAOCAG08AMIIlBqKCAQEAVI0V3jEiOphuxXaoCzELUJG5R7NpbkYTALz17fwZ2Zx49jTfXifmxXFb9yHYCzj40gj9jrsfLUUf51soL2we2V4SnugUG4e(MqZlVyo02wnt7aQoBwlURzLghH3mJD7fUST07uACxQR38L6y8nMNq8msvAHV6nxKyCz6qy9NqEgbzAK4sdg+1qDF/p23mNz1ZzmfqjA0bo/FKWKb9byas8ddslj9zScidFTSGElzr+y1dtCqB8Rfrca3s5ZEIMEp7tUhGN2zpJQ2tHpmvu41Y016hpA3aJSU7hbqL3qtwsJB8xnMLRHfuHKWcWE3tdSEwrjGf4QIDAQAB;
mail._domainkey.jimmylan.fr.	0	DKIM	

OK



Vérifier la configuration DNS

```
# Vérifier l'enregistrement MX  
dig jimmylan.fr MX +short  
  
# Vérifier l'enregistrement A  
dig mail.jimmylan.fr A +short  
  
# Vérifier le PTR (reverse DNS)  
dig -x 203.0.113.10 +short
```



🔥 Configuration du firewall

Avant d'installer Postfix, comprenons les ports que nous allons utiliser !



Comprendre les ports mail

Avant d'ouvrir les ports, il est crucial de comprendre leur rôle.

Dans cette formation, nous allons configurer un serveur mail complet :

- **Postfix** pour envoyer/recevoir des emails
- **Dovecot** pour permettre aux clients (Outlook, Thunderbird, Apple Mail) de consulter leurs emails
- **TLS/SSL** pour sécuriser toutes les communications

Pour cela, nous devons ouvrir plusieurs ports selon leur usage :

- Ports **SMTP** : pour l'envoi de mails
- Ports **IMAP** : pour la lecture de mails
- Ports **POP3** : pour le téléchargement de mails (optionnel)

Ports SMTP - Envoi de mails

Port	Protocole / Mode	Usage	Notes
25	SMTP (plain)	Envoi serveur → serveur	Port standard pour le trafic entre serveurs mail. Les FAI bloquent souvent ce port pour les clients.
587	SMTP Submission + STARTTLS	Envoi client → serveur	Port recommandé pour tous les clients modernes (Outlook, Thunderbird, Apple Mail). TLS négocié après connexion.
465	SMTPS (SSL direct)	Envoi client → serveur	SSL/TLS dès la connexion. Supporté par la plupart des clients mail.

Détails des ports SMTP

Port 25 - SMTP (Communication serveur à serveur)

- C'est le port historique du protocole SMTP
- Utilisé pour le **relay entre serveurs mail** (exemple : gmail.com → votre-domaine.com)
- Les clients finaux (Outlook, Apple Mail) ne doivent **PAS** utiliser ce port
- Beaucoup de FAI bloquent ce port pour éviter le spam

Port 587 - Submission (Client à serveur)

- **Port recommandé** pour l'envoi depuis un client mail
- Obligatoire pour Thunderbird, Apple Mail, et la plupart des clients modernes
- Utilise **STARTTLS** : connexion en clair puis passage en TLS
- Nécessite généralement une authentification (SMTP AUTH)

Port 465 - SMTPS (Client à serveur avec SSL)

- SSL/TLS est actif **dès le début** de la connexion
- Port historique mais toujours très utilisé
- Supporté par Outlook, Apple Mail, Gmail
- Alternative au port 587



 Ports IMAP - Lecture de mails

Port	Protocole / Mode	Usage	Notes
143	IMAP + STARTTLS	Lecture mail depuis client	TLS négocié après connexion via STARTTLS. Port standard IMAP.
993	IMAPS (SSL direct)	Lecture mail sécurisée	SSL/TLS dès la connexion. Port recommandé pour tous les clients modernes.



Détails des ports IMAP

Port 143 - IMAP (avec STARTTLS)

- Port standard du protocole IMAP (Internet Message Access Protocol)
- Permet de consulter ses emails depuis n'importe où
- **STARTTLS** : connexion en clair puis passage en TLS
- Les emails restent sur le serveur (contrairement à POP3)

Port 993 - IMAPS (SSL direct)

- **Port recommandé** pour tous les clients modernes
- SSL/TLS actif **dès le début** de la connexion
- Standard utilisé par défaut par Outlook, Apple Mail, Thunderbird
- Plus sécurisé que le port 143

 **IMAP vs POP3** : IMAP synchronise les emails (ils restent sur le serveur), POP3 les télécharge (et les supprime généralement du serveur).



Ports POP3 - Téléchargement de mails

Port	Protocole / Mode	Usage	Notes
110	POP3 + STARTTLS	Téléchargement depuis serveur	TLS peut être négocié via STARTTLS.
995	POP3S (SSL direct)	Téléchargement sécurisé	SSL/TLS dès la connexion. Port recommandé pour POP3.



Détails des ports POP3

Port 110 - POP3 (avec STARTTLS)

- Port standard du protocole POP3 (Post Office Protocol v3)
- Télécharge les emails du serveur vers le client
- Par défaut, **supprime les emails du serveur** après téléchargement
- **STARTTLS** : connexion en clair puis passage en TLS

Port 995 - POP3S (SSL direct)

- SSL/TLS actif **dès le début** de la connexion
- Port recommandé si vous utilisez POP3
- Certains clients anciens préfèrent POP3 à IMAP

⚠ Note : Dans cette formation, nous privilégierons **IMAP** (plus moderne et pratique), mais nous configurerons quand même POP3 pour la compatibilité.

Résumé rapide des ports

Communication serveur à serveur :

- Port **25** → SMTP (relay entre serveurs mail)

Envoi depuis un client mail (Outlook, Apple Mail, Thunderbird) :

- Port **587** → SMTP Submission avec STARTTLS  **Recommandé**
- Port **465** → SMTPS avec SSL direct

Lecture de mails (IMAP) :

- Port **143** → IMAP avec STARTTLS
- Port **993** → IMAPS avec SSL direct  **Recommandé**

Téléchargement de mails (POP3) :

- Port **110** → POP3 avec STARTTLS
- Port **995** → POP3S avec SSL direct  **Recommandé**

Astuces mnémotechniques

Retenir les ports facilement

Ports pairs = SSL/TLS direct

- **465** (SMTPS) - SSL dès la connexion
- **993** (IMAPS) - SSL dès la connexion
- **995** (POP3S) - SSL dès la connexion

Ports impairs = STARTTLS possible

- **25** (SMTP) - Connexion en clair, STARTTLS optionnel
- **587** (Submission) - Connexion en clair, puis STARTTLS
- **143** (IMAP) - Connexion en clair, puis STARTTLS
- **110** (POP3) - Connexion en clair, puis STARTTLS

Règle simple : Si le port est pair, c'est du SSL direct. Si le port est impair, c'est STARTTLS.

🔧 Configuration importante : master.cf

⚠ ATTENTION : Pour que le port **587** (Submission) fonctionne, il faut activer le service dans `/etc/postfix/master.cf` .

Par défaut, le service `submission` est commenté !

Nous allons le configurer dans ce module pour permettre l'envoi d'emails depuis les clients (Outlook, Thunderbird, Apple Mail) :

- Nous décommenterons les lignes `submission` dans `master.cf`
- Nous activerons **SASL** pour l'authentification
- Nous configurerons **TLS optionnel** (sera rendu obligatoire au module 09)

Sans cette configuration, **Outlook et les autres clients ne pourront pas envoyer d'emails** via votre serveur !

🔥 Ouverture des ports firewall

Maintenant que nous comprenons les ports, ouvrons-les !

```
# Pour Ubuntu (UFW)
sudo ufw allow 25/tcp      # SMTP (serveur à serveur)
sudo ufw allow 587/tcp     # Submission (client à serveur avec STARTTLS)
sudo ufw allow 465/tcp     # SMTPS (client à serveur avec SSL)
sudo ufw allow 143/tcp      # IMAP (lecture mail avec STARTTLS)
sudo ufw allow 993/tcp      # IMAPS (lecture mail avec SSL)
sudo ufw allow 110/tcp      # POP3 (téléchargement avec STARTTLS)
sudo ufw allow 995/tcp      # POP3S (téléchargement avec SSL)
```

```
# Pour Rocky Linux (firewalld)
sudo firewall-cmd --permanent --add-service=smtp           # Port 25
sudo firewall-cmd --permanent --add-service=smtp-submission # Port 587
sudo firewall-cmd --permanent --add-service=smt�           # Port 465
sudo firewall-cmd --permanent --add-service=imap           # Port 143
sudo firewall-cmd --permanent --add-service=imaps          # Port 993
sudo firewall-cmd --permanent --add-service=pop3           # Port 110
sudo firewall-cmd --permanent --add-service=pop3s          # Port 995
sudo firewall-cmd --reload
```

 **Note :** Même si nous n'utiliserons pas beaucoup POP3, nous ouvrons les ports pour assurer la compatibilité avec tous les clients mail.

Installation de Postfix

📦 Installation sur Ubuntu/Debian

```
# Mise à jour des paquets  
sudo apt update && sudo apt upgrade -y  
  
# Installation de Postfix  
sudo apt install postfix -y
```

Pendant l'installation, un assistant graphique apparaît :

1. Type de configuration : Choisissez "Internet Site"

Il se peut que vous ne voyez pas cette étape, car Postfix est déjà installé sur votre système ou en fonction de la version il ne le propose pas, pas d'inquiétude.

2. Nom du système de messagerie : Entrez votre domaine (andromed.cloud)

📦 Installation sur Rocky Linux

```
# Mise à jour du système  
sudo dnf update -y  
  
# Installation de Postfix  
sudo dnf install postfix  
  
# Activer et démarrer Postfix  
sudo systemctl enable postfix  
sudo systemctl start postfix
```



✓ Vérifier l'installation

```
# Vérifier le statut de Postfix  
sudo systemctl status postfix  
  
# Vérifier la version installée  
postconf mail_version  
  
# Vérifier que Postfix écoute sur les bons ports  
sudo ss -tlnp | grep master
```

Vous devriez voir le processus `master` écouter sur le port 25.

Si vous ne voyez pas le processus `master` écouter sur le port 25, vérifiez que le firewall est correctement configuré.



Comprendre l'arborescence de Postfix

📁 Les fichiers importants

Configuration : /etc/postfix/main.cf (config principale)

- /etc/postfix/master.cf (processus)

Tables : /etc/postfix/aliases (alias locaux)

- /etc/postfix/virtual (domaines virtuels)
- /etc/postfix/transport (routage)

Files d'attente : /var/spool/postfix/ (incoming, active, deferred, hold, corrupt)

Logs : /var/log/mail.log (Ubuntu/Debian)

- /var/log/maillog (Rocky/Red Hat)

Configuration de base du main.cf

Le fichier `/etc/postfix/main.cf` contient tous les paramètres de configuration.

```
sudo nano /etc/postfix/main.cf
```

Paramètres essentiels

```
# Identité du serveur
myhostname = mail.example.com
mydomain = example.com
myorigin = $mydomain

# Interfaces réseau
inet_interfaces = all # ou localhost pour tests

# Domaines acceptés
mydestination = $myhostname, localhost.$mydomain, localhost, $mydomain

# Réseaux autorisés (JAMAIS 0.0.0.0/0 = open relay!)
#mynetworks = 127.0.0.0/8 [::ffff:127.0.0.0]/104 [::1]/128
```

🔍 Qu'est ce qu'un relayhost ?

Un relayhost est un serveur SMTP qui permet de relayer les emails vers un autre serveur SMTP.

En gros, c'est un serveur SMTP qui permet de relayer les emails vers un autre serveur SMTP, par exemple si vous avez un serveur mail chez votre FAI et que vous voulez envoyer des emails vers un autre serveur mail, vous pouvez configurer le relayhost pour que Postfix relaye les emails vers le serveur mail de votre FAI.

On ne va pas trop s'attarder dessus dans cette formation, mais il faut savoir que c'est une bonne option dans certains cas.

relayhost : Serveur SMTP relais (optionnel)

```
# Pas de relais (envoi direct)
relayhost = 

# Ou via un relais (exemple : serveur de votre entreprise)
relayhost = [smtp.example.com]:587
```

🔒 Paramètres de sécurité de base

home_mailbox : Format de stockage des emails

```
# Format Maildir (recommandé)
home_mailbox = Maildir

# Format mbox (ancien)
home_mailbox = mail
```

Maildir vs mbox ?

- **Maildir** : Un fichier par email, plus sûr, plus rapide
- **mbox** : Tous les emails dans un seul fichier, risque de corruption

Concrètement, chaque utilisateur disposera d'un répertoire Maildir dédié, par exemple : /home/john/Maildir pour l'utilisateur 'john'.

smtpd_banner : Bannière SMTP (ne pas révéler trop d'infos)

```
# Par défaut (affiche la version)
smtpd_banner = $myhostname ESMTP $mail_name (Ubuntu)

# Version sécurisée (masque les détails)
smtpd_banner = $myhostname ESMTP
```

Exemple de configuration minimale

```
# Nom du serveur
myhostname = mail.jimmylan.fr
mydomain = jimmylan.fr
myorigin = $mydomain

# Interfaces réseau
inet_interfaces = all
inet_protocols = ipv4

# Domaines acceptés
mydestination = $myhostname, localhost.$mydomain, localhost, $mydomain

# Réseaux autorisés
#mynetworks = 127.0.0.0/8 [::ffff:127.0.0.0]/104 [::1]/128

# Pas de relais
relayhost =

# Stockage des emails
home_mailbox = Maildir/
# Attention nous allons plus tard utiliser le transport virtual pour les emails, donc nous ne pouvons pas utiliser home_mailbox.

# Bannière sécurisée
smtpd_banner = $myhostname ESMTP

# Limite de taille des messages (50 MB)
message_size_limit = 52428800
```

Appliquer les modifications

Après avoir modifié `main.cf`, il faut recharger la configuration :

```
# Vérifier la syntaxe  
sudo postfix check  
  
# Si pas d'erreur, recharger  
sudo systemctl reload postfix
```

 **Important :** `postfix check` est votre meilleur ami ! Utilisez-le systématiquement.

Configuration du port 587 (Submission)

Maintenant que la configuration de base est en place, activons le port 587 pour permettre aux clients mail (Outlook, Thunderbird, Apple Mail) d'envoyer des emails via notre serveur.

⚠️ Important : Par défaut, le service submission est **commenté** dans master.cf !





Modifier le fichier master.cf

Ouvrez le fichier de configuration des services :

```
sudo nano /etc/postfix/master.cf
```

Recherchez les lignes suivantes (elles sont commentées avec `#`) :

```
#submission inet n - y - - smtpd
# -o syslog_name=postfix/submission
# -o smtpd_tls_security_level=encrypt
```



Décommenter et configurer submission

Décommentez et modifiez ces lignes pour activer le service :

```
submission inet n - y - - smtpd
  -o syslog_name=postfix/submission
  -o smtpd_tls_security_level=may
  -o smtpd_sasl_auth_enable=yes
  -o smtpd_client_restrictions=permit_sasl_authenticated,reject
  -o smtpd_relay_restrictions=permit_sasl_authenticated,reject
```

🔍 Explications de la configuration

```
submission inet n - y - - smtpd
```

- Active le service submission sur le port 587

```
-o syslog_name=postfix/submission
```

- Nom dans les logs pour distinguer du port 25

```
-o smtpd_tls_security_level=may
```

- TLS optionnel (pour l'instant, on le rendra obligatoire au module 09)

```
-o smtpd_sasl_auth_enable=yes
```

- Active l'authentification SASL (obligatoire pour envoyer des mails)

```
-o smtpd_client_restrictions=permit_sasl_authenticated,reject
```

- Autorise uniquement les clients authentifiés

Installation de SASL pour l'authentification

Pour que les clients puissent s'authentifier, il faut installer SASL (Simple Authentication and Security Layer).

Installation sur Ubuntu/Debian

```
sudo apt install libsasl2-2 sasl2-bin libsasl2-modules -y
```

📦 Installation sur Rocky Linux

```
sudo dnf install cyrus-sasl cyrus-sasl-plain -y
```



🔧 Configuration de SASL dans main.cf

Ajoutez ces lignes dans /etc/postfix/main.cf :

```
# Configuration SASL
smtpd_sasl_type = dovecot
smtpd_sasl_path = private/auth
smtpd_sasl_auth_enable = yes
smtpd_sasl_security_options = noanonymous
smtpd_sasl_local_domain = $myhostname
broken_sasl_auth_clients = yes
```

🔍 Explications SASL

```
smtpd_sasl_type = dovecot
```

- Utilise Dovecot pour l'authentification (nous le configurerons plus tard)
- En attendant Dovecot, SASL utilisera les comptes Unix du système

```
smtpd_sasl_path = private/auth
```

- Socket de communication avec Dovecot

```
smtpd_sasl_auth_enable = yes
```

- Active l'authentification SASL

```
smtpd_sasl_security_options = noanonymous
```

- Interdit les connexions anonymes

```
broken_sasl_auth_clients = yes
```

- Incompatibilité avec les vieux clients (Outlook 2003, etc.)



⚠ Configuration temporaire sans Dovecot

Pour tester immédiatement (avant d'installer Dovecot), modifiez temporairement la configuration SASL :

```
# Configuration SASL temporaire (sans Dovecot)
smtpd_sasl_type = cyrus
smtpd_sasl_path = smtpd
smtpd_sasl_auth_enable = yes
smtpd_sasl_security_options = noanonymous
smtpd_sasl_local_domain = $myhostname
broken_sasl_auth_clients = yes
```

⚠ Cette configuration utilise les comptes Unix du système pour l'authentification.

🔧 Configuration de Cyrus SASL (temporaire)

Créez le fichier de configuration SASL :

```
sudo nano /etc/postfix/sasl/smtpd.conf
```

Ajoutez :

```
pwcheck_method: saslauthd  
mech_list: PLAIN LOGIN
```

Démarrer le service saslauthd

```
# Ubuntu/Debian
sudo systemctl enable saslauthd
sudo systemctl start saslauthd
```

```
# Rocky Linux
sudo systemctl enable saslauthd
sudo systemctl start saslauthd
```



Appliquer la configuration

```
# Vérifier la syntaxe  
sudo postfix check  
  
# Recharger Postfix  
sudo systemctl reload postfix  
  
# Vérifier que le port 587 écoute  
sudo ss -tlnp | grep master
```

Vous devriez maintenant voir les ports **25** et **587** !



✓ Vérifier les ports actifs

```
sudo ss -tlnp | grep master
```

Résultat attendu :

```
LISTEN  0  100  0.0.0.0:25  0.0.0.0:*  users:(("master",pid=1234,fd=13))
LISTEN  0  100  0.0.0.0:587  0.0.0.0:*  users:(("master",pid=1234,fd=17))
```

✓ Le port 587 est maintenant actif !

Tester l'authentification SASL

Vérifions que SASL fonctionne :

```
testsaslauthd -u votre_utilisateur -p votre_mot_de_passe
```

Résultat attendu : 0: OK "Success."

 **Note** : Utilisez un compte Unix existant sur votre système pour tester.

Tester l'envoi via le port 587

Test manuel avec telnet/openssl :

```
telnet localhost 587
```

Une fois connecté, tapez :

```
EHLO mail.example.com
```

Vous devriez voir dans la réponse :

```
250-AUTH PLAIN LOGIN  
250-AUTH=PLAIN LOGIN
```

 L'authentification est disponible !

🎯 Récapitulatif de la configuration

À ce stade, nous avons :

- ✓ Postfix installé et configuré
- ✓ Port 25 actif (serveur à serveur)
- ✓ Port 587 actif (clients à serveur)
- ✓ Authentification SASL fonctionnelle
- ✓ Configuration basique (TLS optionnel)

➡️ SOON **Au module 09 (TLS et Sécurité)**, nous sécuriserons tout ça en rendant TLS **obligatoire** sur le port 587 !

➡️ SOON **Au module 14 (Dovecot)**, nous configurerons Dovecot pour une authentification plus robuste et la lecture des emails (IMAP/POP3).

Premier test d'envoi

Envoyer un email de test

Utilisons la commande `mail` (fournie par Postfix) :

```
echo "Test depuis Postfix" | mail -s "Test" root@localhost
```

Vérifier que l'email est arrivé

```
# Lister les emails de root  
sudo ls -la /root/Maildir/new/  
  
# Lire l'email  
sudo cat /root/Maildir/new/*
```

Test avec un vrai email

```
echo "Ceci est un test" | mail -s "Test Postfix" votre@email.com
```

 **Note** : Si la commande `mail` n'existe pas, installez `mailutils` (Ubuntu) ou `mailx` (Rocky).

Commandes utiles de base

🔧 Gestion du service Postfix

```
# Démarrer Postfix  
sudo systemctl start postfix  
  
# Arrêter Postfix  
sudo systemctl stop postfix  
  
# Redémarrer Postfix (arrêt puis démarrage)  
sudo systemctl restart postfix  
  
# Recharger la configuration (sans interruption)  
sudo systemctl reload postfix  
  
# Vérifier le statut  
sudo systemctl status postfix
```

Voir la file d'attente

```
# Afficher tous les messages en attente  
mailq  
# ou  
postqueue -p
```

Supprimer un message de la file

```
# Supprimer un message spécifique  
sudo postsuper -d ID_DU_MESSAGE  
  
# Supprimer tous les messages  
sudo postsuper -d ALL
```

Voir la configuration active

```
# Afficher toute la configuration  
postconf  
  
# Afficher un paramètre spécifique  
postconf myhostname  
  
# Afficher les paramètres non-défaux  
postconf -n
```

Configuration des alias

Les alias permettent de rediriger les emails d'un compte vers un autre.

Le fichier /etc/aliases

```
sudo nano /etc/aliases
```

Contenu typique :

```
# Redirection des comptes systèmes
postmaster: root
webmaster: root
abuse: root

# Redirection de root vers un vrai email
root: admin@example.com
```

Appliquer les alias

Après modification, il faut recompiler la base de données :

```
sudo newaliases
# ou
sudo postalias /etc/aliases
```



✓ Tester un alias

```
# Envoyer un email à postmaster  
echo "Test alias" | mail -s "Test" postmaster  
  
# Vérifier qu'il arrive bien sur le compte redirigé
```



Logs et debugging

Suivre les logs en temps réel

```
# Ubuntu/Debian  
sudo tail -f /var/log/mail.log  
  
# Rocky Linux  
sudo tail -f /var/log/maillog
```

Rechercher dans les logs

```
# Rechercher tous les logs d'un email spécifique  
sudo grep "test@jimmylan.fr" /var/log/mail.log  
  
# Voir les erreurs uniquement  
sudo grep "error\|warning" /var/log/mail.log
```

Activer le mode verbose

Pour avoir plus de détails dans les logs :

```
sudo postconf -e "smtpd_tls_loglevel = 1"  
sudo postconf -e "smtp_tls_loglevel = 1"  
sudo systemctl reload postfix
```

⚠ **Attention :** Le mode verbose génère beaucoup de logs. À utiliser seulement pour le debug !



Sécurisation minimale

Même pour une configuration de base, quelques mesures de sécurité s'imposent.

🚫 Désactiver les commandes dangereuses

```
# Dans main.cf  
disable_vrfy_command = yes
```

La commande VRFY permet de vérifier si une adresse email existe. Les spameurs l'adorent !

📏 Limiter la taille des messages

```
# Limite à 50 MB  
message_size_limit = 52428800  
  
# Limite de la boîte mail (0 = illimité)  
mailbox_size_limit = 0
```

⌚ Limites de temps

```
# Timeout de connexion SMTP  
smtpd_timeout = 300s  
  
# Timeout client SMTP  
smtp_helo_timeout = 60s
```



🔒 Restrictions de base

```
# Rejeter les connexions trop précoces
smtpd_client_restrictions =
    permit_mynetworks,
    reject_unknown_client_hostname

# Vérifier le HELO
smtpd_helo_restrictions =
    permit_mynetworks,
    reject_invalid_helo_hostname,
    reject_non_fqdn_helo_hostname
```



Troubleshooting courant

✗ Problème : Postfix ne démarre pas

Solution 1 : Vérifier les logs

```
sudo journalctl -u postfix -n 50
```

Solution 2 : Vérifier la syntaxe

```
sudo postfix check
```

Solution 3 : Vérifier les permissions

```
sudo postfix set-permissions
```



✗ Problème : Emails ne partent pas

Vérifier la file d'attente :

```
mailq
```

Voir les erreurs :

```
sudo tail -n 100 /var/log/mail.log | grep error
```

Forcer l'envoi :

```
sudo postqueue -f
```



✖ Problème : Port 25 déjà utilisé

Identifier le processus :

```
sudo lsof -i :25
```

Arrêter le service conflictuel :

```
# Si c'est sendmail  
sudo systemctl stop sendmail  
sudo systemctl disable sendmail
```



✗ Problème : Reverse DNS manquant

Vérifier le PTR :

```
dig -x 51.68.224.131 +short
```

Si le PTR est incorrect ou manquant, contactez votre hébergeur. En attendant, vous pouvez utiliser un relais SMTP avec PTR correct.



Checklist de validation

Avant de passer au module suivant, vérifiez que :

- Postfix est installé et démarré
- Le DNS est correctement configuré (A, MX, PTR)
- Le firewall autorise les ports nécessaires
- Vous pouvez envoyer un email local
- La commande `postconf -n` affiche votre configuration
- Les logs sont accessibles et lisibles
- La file d'attente fonctionne (`mailq`)

Exercice pratique

🎯 Exercice 1 : Installation complète

1. Installez Postfix sur votre système
2. Configurez le `main.cf` avec vos paramètres
3. Envoyez un email à root
4. Vérifiez qu'il est bien arrivé

🎯 Exercice 2 : Configuration des alias

1. Créez un alias pour rediriger `contact@` vers votre email
2. Testez l'envoi à `contact@localhost`
3. Vérifiez la réception

🎯 Exercice 3 : Analyse de logs

1. Envoyez plusieurs emails
2. Suivez les logs en temps réel
3. Identifiez les étapes de traitement du message

🎯 Exercice 4 : Docker : Si vous avez vu Docker

1. Créez un conteneur Postfix avec Docker
2. Configurez-le pour accepter les emails sur le port 2525
3. Testez l'envoi depuis l'extérieur du conteneur

Points clés à retenir

💡 Ce qu'il faut retenir

Configuration minimale : myhostname , mydomain , myorigin sont essentiels

- inet_interfaces définit les interfaces d'écoute - mynetworks contrôle qui peut envoyer des emails

Sécurité de base : Ne jamais faire un open relay (mynetworks = 0.0.0.0/0)

- Toujours vérifier le DNS (surtout le PTR) pour les enregistrements A, MX et PTR
- Limiter les tailles de messages

Outils essentiels : postconf (voir et modifier la configuration) - mailq / postqueue -p (voir la file d'attente)

- postsuper (gérer la file d'attente)
- postfix check (vérifier la syntaxe)

Logs : /var/log/mail.log ou /var/log/maillog - Toujours consulter les logs en cas de problème - tail -f est votre ami

Prochaine étape

Vous avez maintenant un Postfix fonctionnel ! 🎉

Dans le prochain module, nous allons plonger dans **l'architecture interne de Postfix** pour comprendre comment tout fonctionne sous le capot.

Module suivant : Architecture et fonctionnement →

QCM - Module 2 : Installation et configuration

Question 1

Quel est le paramètre obligatoire pour définir le nom complet du serveur ?

- A) mydomain
- B) myhostname
- C) myorigin
- D) inet_interfaces

Question 2

Quelle commande permet de recharger la configuration sans couper les connexions ?

- A) systemctl restart postfix
- B) systemctl reload postfix
- C) postfix restart
- D) killall postfix

Question 3

Quel enregistrement DNS est OBLIGATOIRE pour un serveur mail ?

- A) A
- B) CNAME
- C) MX
- D) TXT

Question 4

Quel paramètre Postfix contrôle les interfaces réseau sur lesquelles le service SMTP écoute ?

- A) relayhost
- B) inet_interfaces
- C) smtpd_banner
- D) alias_database

Question 5

Quelle commande vérifie la configuration Postfix pour détecter les erreurs avant un rechargement ?

- A) postfix check
- B) postconf -n
- C) systemctl status postfix
- D) postqueue -p



Réponses - Module 2

Question 1 : Réponse B - `myhostname = mail.example.com` définit le FQDN du serveur. C'est le paramètre le plus important !

Question 2 : Réponse B - `systemctl reload postfix` recharge la config **sans couper** les connexions en cours. `restart` couperait tout !

Question 3 : Réponse C - L'enregistrement **MX** indique le serveur mail responsable du domaine. Sans lui, impossible de recevoir des emails !

Question 4 : Réponse B - `inet_interfaces` définit si Postfix écoute sur `all` , `loopback-only` ou une IP précise. Pratique pour limiter l'écoute à `localhost` sur un serveur relai.

Question 5 : Réponse A - `postfix check` réalise une série de vérifications (permissions, syntaxe, ownership) et affiche les problèmes potentiels avant redémarrage.



Exercice pratique - Module 2

🎯 Objectif

Installer et configurer Postfix pour la première fois

📋 Tâches (15 minutes)

- 1. Installation :** Installez Postfix sur votre système
- 2. Configuration :** Configurez ces paramètres dans `/etc/postfix/main.cf` :

- `myhostname = mail.votredomaine.local`
- `mydomain = votredomaine.local`
- `myorigin = $mydomain`
- `inet_interfaces = all`
- `mydestination = $myhostname, localhost, $mydomain`

3. Test : Envoyez un email de test local :

```
echo "Test Postfix" | mail -s "Test" $USER  
ls ~/Maildir/new/
```



4. **Vérification** : Consultez les logs : `tail -f /var/log/mail.log`

Aide : En cas de problème, utilisez `postfix check`

✓ Réponse :

```
# Installation  
sudo apt install postfix  
  
# Configuration  
sudo nano /etc/postfix/main.cf
```



dans la conf :

```
myhostname = mail.votredomaine.local
mydomain = votredomaine.local
smtpd_banner = $myhostname ESMTP
myorigin = $mydomain
home_mailbox = Maildir/
disable_vrfy_command = yes
inet_interfaces = all
mydestination = $myhostname, localhost, $mydomain
inet_protocols = ipv4
```

```
# Rechargement de la configuration  
sudo systemctl reload postfix  
  
# Test d'envoi  
echo "Test Postfix" | mail -s "Test" $USER  
ls ~/Maildir/new/
```



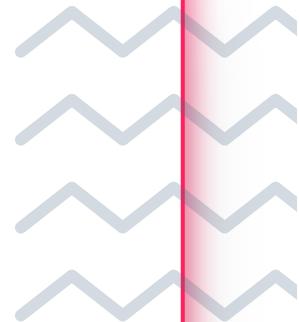
```
# Vérification des logs  
tail -f /var/log/mail.log
```



Architecture et Fonctionnement



Comprendre les entrailles de Postfix



Introduction à l'architecture

Postfix n'est pas un simple programme, c'est une **orchestration** de plusieurs processus indépendants travaillant ensemble.

Pensez à une usine où chaque ouvrier a une tâche précise : c'est exactement comme ça que Postfix fonctionne !

La philosophie de conception

🎯 Principes fondamentaux

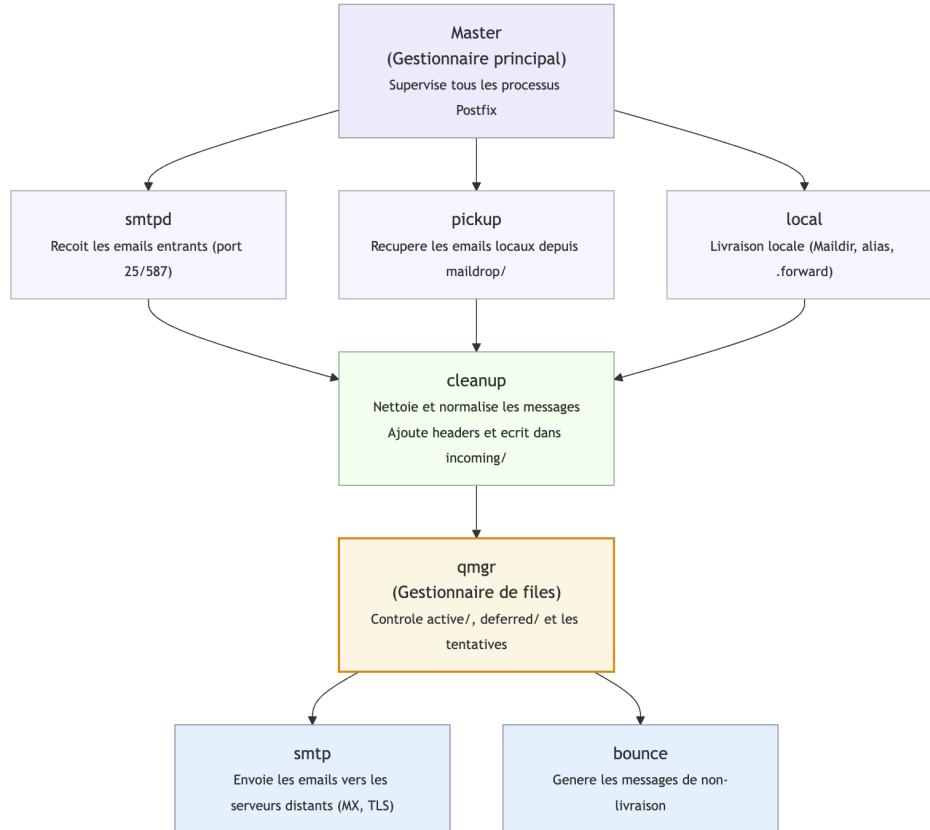
Séparation des priviléges : Chaque processus tourne avec le minimum de droits nécessaires - Si un processus est compromis, les dégâts sont limités

Architecture modulaire : Chaque tâche est gérée par un processus dédié - Facile de remplacer ou désactiver un composant - Isolation des pannes

Communication par files d'attente : Les processus ne se parlent pas directement - Ils communiquent via des fichiers dans des répertoires - Robustesse : si un processus crash, les messages ne sont pas perdus

Vue d'ensemble de l'architecture





Le processus Master

Le **master** est le chef d'orchestre. C'est lui qui lance tous les autres processus, surveille leur santé, les redémarre en cas de crash et gère leur cycle de vie.

Configuration du master

Le fichier `/etc/postfix/master.cf` définit tous les services :

```
# service type  private unpriv  chroot  wakeup  maxproc command
smtp    inet  n      -       y       -       -       smtpd
pickup  unix  n      -       y       60      1       pickup
cleanup  unix  n      -       y       -       0       cleanup
qmgr    unix  n      -       n       300     1       qmgr
```

Si vous voulez utiliser un environnement chrooté, vous pouvez le faire en ajoutant `y` à la place de `-` dans la colonne `chroot`.

exemple :

```
smtp    inet  n      -       y       -       -       smtpd
```

Décryptons une ligne :

```
smtp      inet  n   -   y   -   -   smtpd
```

- **service** : smtp - nom du service
- **type** : inet - socket internet (vs unix pour socket Unix)
- **private** : n - accessible de l'extérieur
- **unpriv** : - - tourne avec les privilèges par défaut
- **chroot** : y - tourne dans un environnement chrooté (isolé)
- **wakeup** : - - pas de réveil automatique
- **maxproc** : - - nombre max de processus (défaut)
- **command** : smtpd - programme à exécuter



Les processus principaux

smtpd (SMTP daemon)

Rôle : Recevoir les emails depuis Internet ou les clients

Responsabilités : Écoute sur le port 25 (ou 587 pour submission) - Dialogue SMTP avec les clients

- Applique les restrictions et politiques
- Accepte ou rejette les messages
- Passe les messages acceptés à cleanup

Analogie : C'est le réceptionniste de l'hôtel qui accueille les clients et vérifie leurs réservations.



Rôle : Récupérer les emails déposés localement

Responsabilités : Surveille le répertoire `maildrop/`

- Récupère les emails déposés par les programmes locaux (via `sendmail`)
- Passe les messages à `cleanup`

Analogie : C'est l'employé qui ramasse le courrier déposé dans la boîte aux lettres interne.

```
pickup    unix  n      -      y      60     1      pickup
```



🧹 cleanup

Rôle : Nettoyer et normaliser les messages

Responsabilités : Ajoute les en-têtes manquants (Date, Message-ID, etc.)

- Complète les adresses (user → user@domain.com)
- Extrait les destinataires des en-têtes
- Écrit le message dans la file incoming/ - Notify le qmgr

Analogie : C'est le service qualité qui vérifie que le courrier est conforme avant expédition.

```
cleanup  unix  n      -     y      -     0      cleanup
```

qmgr (Queue manager)

Rôle : Gérer les files d'attente - C'est le **cœur** de Postfix !

Responsabilités : Surveille les files d'attente

- Décide quand envoyer les messages
- Choisit le bon processus de livraison
- Gère les tentatives et les délais - Optimise l'envoi (regroupe par destination)

Analogie : C'est le chef de gare qui décide quels trains partent, quand, et vers où.

```
qmgr      unix n      -      n      300     1      qmgr
```

[Revenir au sommaire](#)

smtp (SMTP client)

Rôle : Envoyer les emails vers d'autres serveurs

Responsabilités : Se connecte aux serveurs destinataires

- Négocie TLS si possible
- Transmet le message
- Gère les erreurs temporaires (retry) et permanentes - Notifie le `qmgr` du résultat

Analogie : C'est le facteur qui livre le courrier chez le destinataire.



Rôle : Livrer les emails locaux

Responsabilités : Livre les emails dans les boîtes locales

- Gère les fichiers .forward
- Applique les alias
- Peut invoquer des programmes externes (filtres)

Analogie : C'est le facteur qui distribue le courrier dans les boîtes aux lettres de l'immeuble.

smtp inet n - y - - - smtpd





Rôle : Gérer les rebonds (emails non délivrés)

Responsabilités : Génère les messages de non-délivrance (bounce)

- Notifie l'expéditeur en cas d'échec définitif
- Gère les messages d'avertissement (delay warning)

Analogie : C'est le service retour qui renvoie le courrier avec la mention "n'habite pas à l'adresse indiquée".

```
bounce      unix  n      -      n      -      0      bounce
```



trivial-rewrite

Rôle : Réécriture d'adresses

Responsabilités : Résout les adresses (lookup DNS)

- Applique les règles de réécriture
- Détermine le transport approprié

Analogie : C'est le service qui réécrit les adresses, par exemple si vous avez un alias sur votre domaine, Postfix va réécrire l'adresse pour que l'email arrive à la bonne personne.

```
trivial-rewrite unix - - - - - trivial-rewrite
```

Les files d'attente

Postfix utilise plusieurs files d'attente dans /var/spool/postfix/ :

maildrop

Contenu : Messages déposés localement par les programmes

Processus responsable : pickup

Durée de vie : Très courte (quelques secondes)

incoming

Contenu : Messages reçus, en cours de nettoyage

Processus responsable : cleanup

Durée de vie : Courte (secondes à minutes)

active

Contenu : Messages en cours de livraison

Processus responsable : qmgr

Taille limite : Contrôlée (évite la saturation mémoire)

deferred

Contenu : Messages en échec temporaire

Processus responsable : qmgr

Durée de vie : Jusqu'à 5 jours par défaut

Les messages en `deferred` sont retentés selon un algorithme exponentiel : 1ère tentative immédiate, 2ème après quelques minutes, 3ème après 15-30 minutes, 4ème après 1 heure, etc.



Contenu : Messages mis en attente manuellement

Processus responsable : Admin (vous !)

Durée de vie : Jusqu'à libération manuelle



Contenu : Messages corrompus

Processus responsable : Aucun (pour investigation)

Durée de vie : Jusqu'à suppression manuelle

Le parcours d'un email

Email entrant (réception)

1. smtpd reçoit la connexion (port 25)
↓
2. Applique les restrictions (RBL, SPF...)
↓
3. cleanup normalise le message
↓
4. qmgr place en queue active
↓
5. local délivre dans Maildir/
✓ Email livré !

✉ Email sortant (envoi)

1. Application → sendmail → maildrop/
↓
2. pickup récupère → cleanup
↓
3. qmgr place en queue active
↓
4. smtp se connecte au serveur distant
↓
5. Transmission via SMTP
✓ Email envoyé !

✗ Email en échec

1. smtp ne peut pas livrer (erreur connexion)
↓
2. qmgr → deferred/ (file différée)
↓
3. Nouvelles tentatives espacées (5min, 15min, 1h...)
↓
4. Après 5 jours → bounce (NDR à l'expéditeur)
■ Message abandonné



Communication

Les processus communiquent via :

- **Sockets Unix** dans /var/spool/postfix/
- **Fichiers** dans les files d'attente
- **Verrous (locks)** pour éviter la corruption

 C'est transparent, pas besoin de configurer !

Modes de fonctionnement

💡 Mode "null client"

Postfix configuré uniquement pour **envoyer**, pas recevoir :

```
inet_interfaces = loopback-only
mydestination =
relayhost = [smtp.jimmylan.fr]
```



Cas d'usage :

- Serveur web qui envoie des notifications
- **Application qui ne reçoit jamais d'emails** ⇒ le fameux "**Ne répondez pas à cet email**"
- Machine dans un LAN sans exposition Internet

🌐 Mode "Internet site"

Configuration classique, reçoit et envoie :

```
inet_interfaces = all
mydestination = $myhostname, $mydomain, localhost
mynetworks = 127.0.0.0/8
```

Mode "relay"

Relais entre réseaux :

```
inet_interfaces = all
relay_domains = $mydestination, domain1.com, domain2.com
relayhost = [mail.backend.com]
```

Sécurité par conception



La plupart des processus tournent dans un chroot :

Le **chroot** est un mécanisme de sécurité qui enferme un processus dans un "enclos" dédié, empêchant tout accès au reste du système hôte. On peut l'imaginer comme une cage vitrée dans un laboratoire : même si le processus essaie de s'échapper, il ne pourra toucher qu'à ce qui se trouve dans son espace limité, protégeant ainsi le véritable environnement du serveur.

```
/var/spool/postfix/ # Racine du chroot
└── etc/           # Fichiers de config nécessaires
└── lib/           # Bibliothèques
└── usr/
    └── lib/
        [process directories]
```

Le processus ne peut pas accéder à `/etc/passwd` , `/root` , etc.

Même si compromis, les dégâts sont limités au chroot !

🔒 Séparation des priviléges

Processus	Utilisateur	Droits
master	root	Supervision uniquement
smtpd	postfix	Lecture réseau
cleanup	postfix	Écriture queue
qmgr	postfix	Gestion queue
local	root	Écriture mailbox (nécessaire)
smtp	postfix	Écriture réseau

Même si `smtpd` est compromis, l'attaquant ne peut pas :

- Modifier les fichiers de config
- Lire les mailbox des utilisateurs
- Écrire ailleurs que dans la queue

Processus auxiliaires



Rôle : Compteur de connexions

Surveille et limite :

- Nombre de connexions par IP
- Taux de connexions
- Détection de comportements suspects



Rôle : Proxy pour les lookups (DB, LDAP)

Centralise les connexions aux bases externes :

- Évite trop de connexions simultanées
- Cache les résultats
- Optimise les performances



Rôle : Proxy TLS

Gère les connexions TLS :

- Négociation SSL/TLS
- Vérification des certificats
- Chiffrement/déchiffrement



Rôle : Logger centralisé

Reçoit les logs de tous les processus chrootés et les écrit dans syslog.

Visualiser l'architecture en action

Voir les processus actifs

ps aux | grep postfix

Vous devriez voir :

```
root      postfix  master
postfix   postfix  qmgr
postfix   postfix  pickup
postfix   postfix  cleanup
postfix   postfix  smtpd
...
...
```



Surveiller les connexions

```
sudo postfix status  
# Connexions actives sur le port 25  
sudo ss -tnp | grep :25
```

Voir l'activité des queues

```
watch -n 1 'mailq | head -n 20'
```

Debugging de l'architecture

🐛 Mode verbose

Activer les logs détaillés :

```
sudo postconf -e "debug_peer_list = example.com"  
sudo postconf -e "debug_peer_level = 2"  
sudo systemctl reload postfix
```

Suivre un message spécifique

Dans les logs, chaque message a un **Queue ID** unique :

```
postfix/smtpd[1234]: ABC123: client=example.com[1.2.3.4]
postfix/cleanup[1235]: ABC123: message-id=<test@example.com>
postfix/qmgr[1236]: ABC123: from=<sender@example.com>, size=1234
postfix/smtp[1237]: ABC123: to=<dest@example.com>, status=sent
```

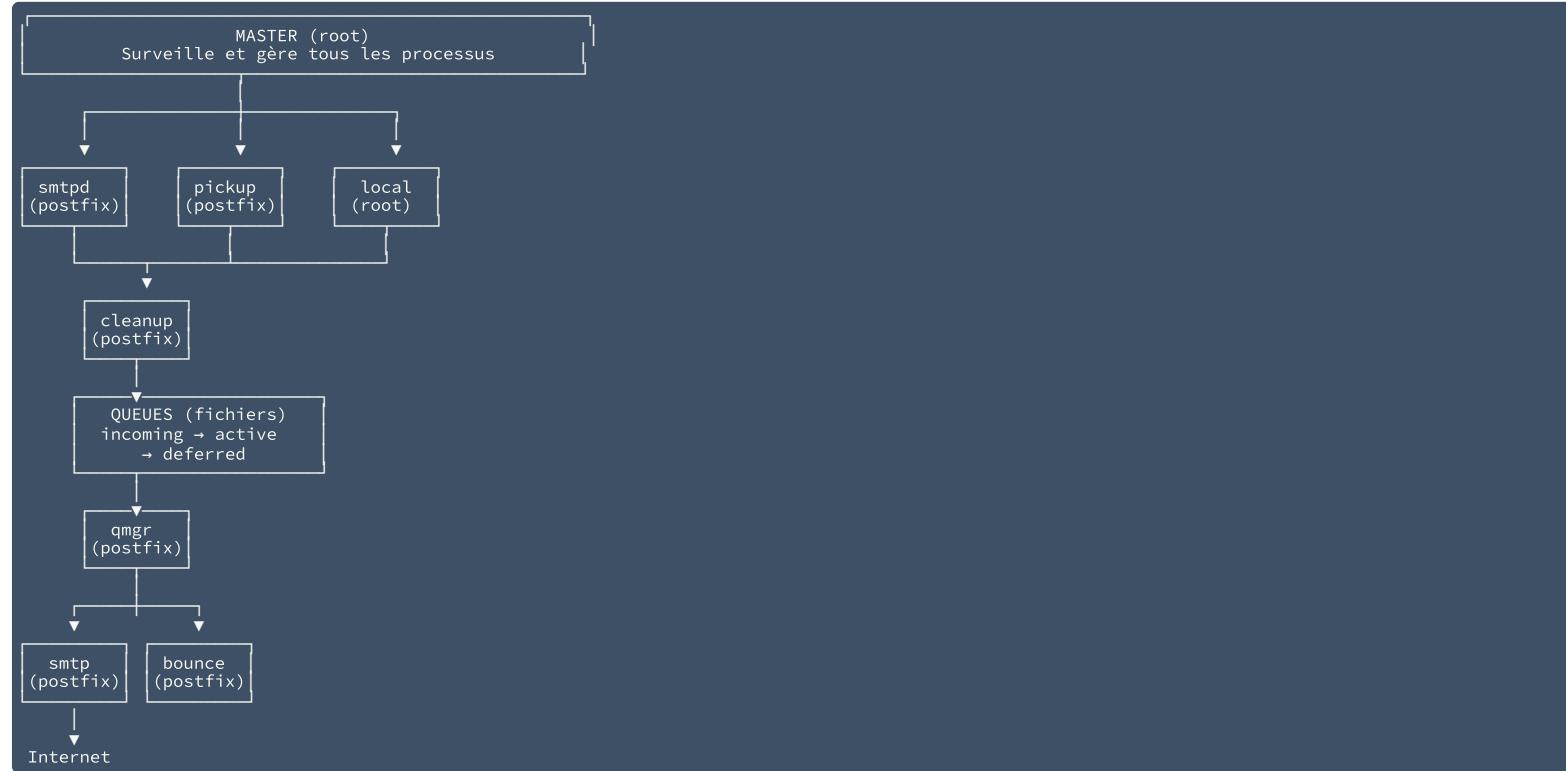
Pour suivre le message ABC123 :

```
sudo grep ABC123 /var/log/mail.log
```

Statistiques en temps réel

```
# Statistiques globales  
sudo postqueue -p | tail -n 1  
  
# Statistiques par destination  
sudo qshape active  
sudo qshape deferred
```

Schéma récapitulatif complet



Points clés à retenir

Architecture modulaire :

- Master = chef d'orchestre
- Processus spécialisés : smtpd, smtp, qmgr, cleanup, local
- Communication via sockets et fichiers



Files d'attente :

- mailldrop → incoming → active → livraison
- deferred pour les échecs
- hold pour mise en attente manuelle



Sécurité :

- Séparation des privilèges
- Chroot pour isoler les processus



Exercices pratiques

Exercice 1 : Observer les processus

```
ps aux | grep postfix  
sudo postconf -d | grep default_process_limit
```

🎯 Exercice 2 : Suivre un message

```
echo "Test" | mail -s "Suivi" user@example.com  
sudo tail -f /var/log/mail.log | grep "Queue ID"
```

Prochaine étape

Maintenant que vous comprenez **comment** Postfix fonctionne, nous allons plonger dans le **fichier de configuration principale** : main.cf

Module suivant : Configuration du main.cf →



QCM - Module 3 : Architecture de Postfix

Question 1

Quel processus est le "chef d'orchestre" de Postfix ?

- A) smtpd
- B) qmgr
- C) master
- D) cleanup

Question 2

Quel processus gère les files d'attente et décide quand envoyer les messages ?

- A) smtpd
- B) qmgr (queue manager)
- C) pickup
- D) smtp



Question 3

Dans quelle file d'attente se trouvent les messages en cours de livraison ?

- A) maildrop
- B) incoming
- C) active
- D) deferred

Question 4

Quel processus Postfix accepte les connexions SMTP entrantes depuis les clients ou d'autres MTAs ?

- A) pickup
- B) smtpd
- C) smtp
- D) cleanup



Question 5

Quel est le rôle du processus `pickup` dans l'architecture Postfix ?

- A) Envoyer les messages sortants vers Internet
- B) Nettoyer les en-têtes et appliquer les règles MIME
- C) Reprendre les messages déposés dans `maildrop` par les agents locaux
- D) Vérifier les signatures DKIM

Réponses - Module 3

Question 1 : Réponse C - Le processus **master** est le chef d'orchestre. Il lance tous les autres processus, surveille leur santé et les redémarre si nécessaire.

Question 2 : Réponse B - **qmgr** (queue manager) est le CŒUR de Postfix. Il gère toutes les files d'attente et décide quand envoyer les messages.

Question 3 : Réponse C - La file **active** contient les messages prêts à être envoyés et en cours de livraison (max 20 000 par défaut).

Question 4 : Réponse B - **smtpd** est le démon serveur SMTP. Il gère l'authentification, applique les restrictions et remet les messages au processus **cleanup**.

Question 5 : Réponse C - **pickup** surveille la file **maildrop** (messages injectés localement par **sendmail** / **postdrop**) et les transfère vers **cleanup** pour traitement.

Exercice pratique - Module 3

🎯 Objectif

Observer l'architecture de Postfix en action

📋 Tâches (10 minutes)

1. Observer les processus :

```
ps aux | grep postfix  
pstree -p $(pgrep -o master)
```

2. Suivre un message :

```
# Envoyer un email  
echo "Test" | mail -s "Test architecture" $USER  
  
# Suivre dans les logs  
tail -f /var/log/mail.log | grep "postfix"
```

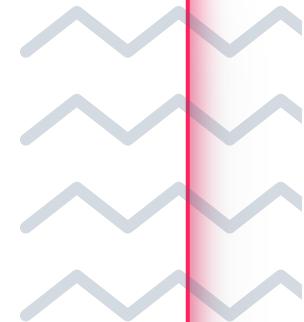
3. Explorer les queues :

```
sudo ls -la /var/spool/postfix/active/  
sudo postqueue -p
```



Bonus : Utilisez `postcat -q QUEUE_ID` pour lire un message





Configuration du main.cf



Maîtriser le fichier de configuration principal de Postfix

Introduction au main.cf

Le fichier `/etc/postfix/main.cf` est le **cerveau** de Postfix. C'est là que tout se configure !



Structure du fichier

Format

Très simple :

```
# Commentaire
paramètre = valeur

# Valeurs multiples
paramètre = valeur1,
            valeur2,
            valeur3
```

🔍 Règles de syntaxe

Commentaires : # texte

- **Continuation** : indentation ou backslash
- **Variables** : \$mydomain
- **Listes** : virgules ou espaces

```
# Commentaire
myhostname = mail.example.com  # Commentaire fin de ligne

# Continuation avec indentation
smtpd_recipient_restrictions =
    permit_mynetworks,
    reject_unauth_destination

# Variables
mydomain = example.com
myorigin = $mydomain

# Listes
mydestination = $myhostname, localhost, $mydomain
```

[Revenir au sommaire](#)

Modifier le fichier

```
# Éditer  
sudo nano /etc/postfix/main.cf  
  
# Vérifier la syntaxe  
sudo postfix check  
  
# Appliquer les changements  
sudo systemctl reload postfix
```



Paramètres d'identité

myhostname

Le nom complet de votre serveur (FQDN - Fully Qualified Domain Name)

```
myhostname = mail.example.com
```

Important : Doit correspondre au PTR (reverse DNS) - Utilisé dans les en-têtes des emails - Première chose que voient les autres serveurs



Le nom de votre domaine

```
mydomain = example.com
```

Par défaut, Postfix déduit mydomain depuis myhostname :

```
myhostname = mail.example.com  
→ mydomain = example.com (automatique)
```





Le domaine qui apparaît dans le champ From: des emails locaux

```
myorigin = $mydomain
```

Exemple : Sans myorigin , un email de root apparaîtrait comme root@mail.example.com . Avec myorigin = \$mydomain , il apparaît comme root@example.com (plus propre !)

Paramètres réseau

inet_interfaces

Interfaces réseau sur lesquelles Postfix écoute

```
# Écouter partout (défaut pour serveur mail)
inet_interfaces = all

# Seulement localhost (pour tests)
inet_interfaces = localhost

# Interfaces spécifiques
inet_interfaces = 192.168.1.10, 127.0.0.1
```

Pourquoi choisir 192 par exemple ?

192.168.1.10 est l'adresse IP de votre serveur.

Donc on dit que le serveur écoute sur l'interface 192.168.1.10 et sur l'interface 127.0.0.1.

Cela limite l'exposition du service et améliore la sécurité si le serveur possède plusieurs interfaces ou adresses IP.

Attention : Si vous changez vers `all` , assurez-vous que votre firewall est configuré !



inet_protocols

Protocoles IP supportés

```
# IPv4 et IPv6 (défaut)
inet_protocols = all

# IPv4 seulement
inet_protocols = ipv4

# IPv6 seulement
inet_protocols = ipv6
```

En 2025, `all` est recommandé, mais si vous n'avez pas d'IPv6 configuré, mettez `ipv4` pour éviter des warnings.



proxy_interfaces

Adresses IP externes (si derrière un NAT/proxy)

```
# Si votre serveur est en 192.168.1.10 mais exposé en 203.0.113.10
proxy_interfaces = 203.0.113.10
```

Postfix considère ces adresses comme "locales" même si elles ne sont pas directement sur ses interfaces.

Nous n'allons pas utiliser cette option dans ce cours.

Paramètres de destination

mydestination

Domaines pour lesquels Postfix accepte les emails comme destination finale

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

Exemples :

```
# Serveur mail classique
mydestination = mail.example.com, example.com, localhost

# Null client (n'accepte rien de l'extérieur)
mydestination =

# Plusieurs domaines
mydestination = example.com, example.org, localhost
```

Attention : Ne confondez pas mydestination et relay_domains !

- mydestination : Postfix **livre localement**
- relay_domains : Postfix **relaie ailleurs**

relay_domains

Domaines pour lesquels Postfix accepte de relayer les emails

```
# Pas de relais (défaut)
relay_domains = 

# Relais pour certains domaines
relay_domains = subsidiary.example.com, partner.com
```

Important : Attention aux open relays !

```
# ❌ NE JAMAIS FAIRE ÇA
relay_domains = *
```

Votre serveur deviendrait un relais ouvert, utilisable par tous les spameurs du monde !

Donc dans notre cas nous allons simplement laisser la valeur par défaut.

```
mydestination = $myhostname, localhost.$mydomain, localhost, $mydomain
relay_domains = $mydestination
```



Réseaux autorisés à envoyer des emails sans authentification

```
# Seulement localhost (recommandé)
mynetworks = 127.0.0.0/8 [::ffff:127.0.0.0]/104 [::1]/128

# Réseau local en plus
mynetworks = 127.0.0.0/8, 192.168.1.0/24

# Détection automatique (déconseillé)
mynetworks_style = subnet
```

Méthode de détection :

```
# host : Seulement la machine locale
mynetworks_style = host

# subnet : Tout le sous-réseau (dangereux !)
mynetworks_style = subnet

# class : Toute la classe réseau (très dangereux !)
mynetworks_style = class
```

 **Bonne pratique :** Spécifiez toujours mynetworks manuellement, ne laissez pas Postfix deviner.



Serveur SMTP à utiliser pour envoyer tous les emails sortants

```
# Pas de relais (envoi direct)
relayhost =

# Via un serveur SMTP
relayhost = [smtp.example.com]

# Via un serveur avec port spécifique
relayhost = [smtp.example.com]:587
```

Les crochets [] : Désactivent le lookup MX

Sans crochets : relayhost = smtp.example.com → Postfix cherche l'enregistrement MX de smtp.example.com

Avec crochets : relayhost = [smtp.example.com] → Postfix se connecte directement à smtp.example.com

Cas d'usage : Serveur derrière un FAI qui bloque le port 25 - Application qui envoie via le serveur mail de l'entreprise - Serveur avec IP blacklistée qui passe par un relais propre

Paramètres de stockage

📁 home_mailbox

Format de stockage des emails locaux

```
# Format Maildir (un fichier par email)
home_mailbox = Maildir/
# Format mbox (tous les emails dans un fichier)
home_mailbox = mail/
```

Maildir est recommandé car : Plus sûr (pas de corruption d'un fichier énorme) - Plus rapide (accès concurrent possible) - Compatible avec IMAP - Standard moderne

message_size_limit

Taille maximum d'un message (headers + body)

```
# 50 MB (défaut : 10 MB)
message_size_limit = 52428800

# Illimité (déconseillé)
message_size_limit = 0
```

Calcul : $10 \text{ MB} = 10 * 1024 * 1024 = 10485760 \text{ bytes}$ - $50 \text{ MB} = 50 * 1024 * 1024 = 52428800 \text{ bytes}$

mailbox_size_limit

Taille maximum d'une mailbox (format mbox uniquement)

```
# Illimité (pour Maildir)
mailbox_size_limit = 0

# 1 GB pour mbox
mailbox_size_limit = 1073741824
```

Avec Maildir, ce paramètre n'a pas de sens (chaque email est un fichier séparé).

Paramètres de bannière et identification

🎭 smtpd_banner

Message affiché lors de la connexion SMTP

```
# Par défaut (révèle la version)
smtpd_banner = $myhostname ESMTP $mail_name (Ubuntu)

# Version sécurisée (masque les détails)
smtpd_banner = $myhostname ESMTP

# Personnalisée
smtpd_banner = $myhostname ESMTP - No spam please
```

Sécurité : Ne révélez pas votre version de Postfix !

	220 mail.example.com ESMTP Postfix (Ubuntu 3.6.4)
	220 mail.example.com ESMTP

⌚ delay_warning_time

Délai avant d'envoyer un avertissement à l'expéditeur si l'email n'est pas encore délivré

```
# Avertissement après 4 heures (défaut)
delay_warning_time = 4h

# Désactiver les avertissements
delay_warning_time = 0h
```

⌚ maximal_queue_lifetime

Durée maximum qu'un message peut rester en file d'attente

```
# 5 jours (défaut)
maximal_queue_lifetime = 5d

# 1 jour
maximal_queue_lifetime = 1d
```

Après ce délai, un email de bounce est envoyé à l'expéditeur et le message est supprimé.

⌚ **bounce_queue_lifetime**

Durée maximum pour les messages de bounce

```
# 5 jours (défaut)  
bounce_queue_lifetime = 5d  
  
# 1 jour (bounce abandonnés plus vite)  
bounce_queue_lifetime = 1d
```

Paramètres de restrictions

🚫 `smtpd_recipient_restrictions`

Restrictions sur les destinataires (qui peut recevoir des emails)

```
smtpd_recipient_restrictions =  
    permit_mynetworks,  
    permit_sasl_authenticated,  
    reject_unauth_destination,  
    reject_invalid_hostname,  
    reject_non_fqdn_recipient,  
    reject_unknown_recipient_domain
```

Ordre important ! Les règles sont évaluées de haut en bas.

Décryptons :

1. permit_mynetworks : Autorise les IPs de mynetworks
2. permit_sasl_authenticated : Autorise les utilisateurs authentifiés
3. reject_unauth_destination : Rejette si destination pas dans mydestination ou relay_domains
4. reject_invalid_hostname : Rejette les hostnames invalides
5. reject_non_fqdn_recipient : Rejette si destinataire pas en FQDN
6. reject_unknown_recipient_domain : Rejette si le domaine destinataire n'existe pas

⚠ CRUCIAL : `reject_unauth_destination` doit **toujours** être présent !

Sans cette règle, votre serveur devient un open relay.

🚫 `smtpd_sender_restrictions`

Restrictions sur les expéditeurs

```
smtpd_sender_restrictions =  
    permit_mynetworks,  
    permit_sasl_authenticated,  
    reject_non_fqdn_sender,  
    reject_unknown_sender_domain
```

👋 smtpd_helo_restrictions

Restrictions sur la commande HELO/EHLO

```
smtpd_helo_restrictions =  
    permit_mynetworks,  
    reject_invalid_helo_hostname,  
    reject_non_fqdn_helo_hostname,  
    reject_unknown_helo_hostname
```

Pourquoi c'est important ?

Beaucoup de spammers envoient des HELO invalides :

```
HELO localhost
HELO 192.168.1.1
HELO [203.0.113.10]
```

Ces règles les bloquent !

⚡ smtpd_client_restrictions

Restrictions sur les clients qui se connectent

```
smtpd_client_restrictions =  
    permit_mynetworks,  
    reject_unknown_client_hostname,  
    check_client_access hash:/etc/postfix/client_access
```

Paramètres d'authentification

🔒 smtpd_sasl_auth_enable

Activer l'authentification SASL pour l'envoi d'emails

```
smtpd_sasl_auth_enable = yes
```

SASL = Simple Authentication and Security Layer

Permet aux clients d'envoyer des emails après authentification (username + password).

🔑 smtpd_sasl_type

Type de mécanisme SASL

```
# SASL basique (pour cette formation)
# smtpd_sasl_type = cyrus
# smtpd_sasl_path = smtpd

# Cyrus SASL
smtpd_sasl_type = cyrus
```

🛡 **smtpd_sasl_security_options**

Options de sécurité SASL

```
# Désactiver les méthodes anonymes  
smtpd_sasl_security_options = noanonymous  
  
# En TLS, autoriser les méthodes plain  
smtpd_sasl_tls_security_options = noanonymous
```



smtpd_sasl_local_domain

Domaine SASL local

```
smtpd_sasl_local_domain = $mydomain
```

Paramètres TLS

🔒 smtpd_tls_cert_file

Chemin vers le certificat SSL

```
smtpd_tls_cert_file = /etc/letsencrypt/live/mail.example.com/fullchain.pem
```



🔑 **smtpd_tls_key_file**

Chemin vers la clé privée SSL

```
smtpd_tls_key_file = /etc/letsencrypt/live/mail.example.com/privkey.pem
```

🔒 **smtpd_tls_security_level**

Niveau de sécurité TLS pour les connexions entrantes

```
# Pas de TLS (déconseillé)
smtpd_tls_security_level = none

# TLS si possible (opportuniste)
smtpd_tls_security_level = may

# TLS obligatoire pour tous
smtpd_tls_security_level = encrypt

# TLS obligatoire + vérification certificat
smtpd_tls_security_level = verify
```

En 2025, utilisez au minimum `may`, idéalement `encrypt` pour le port 587.

smtp_tls_security_level

Niveau de sécurité TLS pour les connexions sortantes

```
# TLS si le serveur distant le supporte
smtp_tls_security_level = may

# TLS obligatoire
smtp_tls_security_level = encrypt
```



smtpd_tls_protocols

Versions de TLS acceptées

```
# TLS 1.2 et 1.3 uniquement (recommandé en 2025)
smtpd_tls_protocols = >TLSv1.2

# TLS 1.3 uniquement (très strict)
smtpd_tls_protocols = >TLSv1.3
```

⚠️ Important : Désactivez TLS 1.0 et 1.1, ils sont obsolètes et vulnérables !

smtpd_tls_loglevel

Niveau de logging TLS

```
# Minimal (défaut)
smtpd_tls_loglevel = 0

# Debug (pour troubleshooting)
smtpd_tls_loglevel = 1

# Très verbeux
smtpd_tls_loglevel = 2
```

Paramètres de file d'attente

queue_run_delay

Fréquence de traitement de la file d'attente

```
# Toutes les 5 minutes (défaut)
queue_run_delay = 300s

# Plus rapide (toutes les minutes)
queue_run_delay = 60s
```

12 34 minimal_backoff_time

Délai minimum avant de retenter l'envoi

```
# 5 minutes (défaut)
minimal_backoff_time = 300s

# 1 minute
minimal_backoff_time = 60s
```

12 34 maximal_backoff_time

Délai maximum entre deux tentatives

```
# 4000 secondes (défaut)
maximal_backoff_time = 4000s

# 1 heure
maximal_backoff_time = 3600s
```

Algorithm des retries

Postfix utilise un algorithme **exponentiel** :

```
Tentative 1 : immédiat  
Tentative 2 : après minimal_backoff_time (5 min)  
Tentative 3 : 10 min  
Tentative 4 : 20 min  
Tentative 5 : 40 min  
...  
Jusqu'à maximal_backoff_time
```

Paramètres de performance

⚡ `default_process_limit`

Nombre maximum de processus Postfix simultanés

```
# 100 (défaut)
default_process_limit = 100

# Pour serveur haute capacité
default_process_limit = 500
```

qmgr_message_active_limit

Nombre maximum de messages actifs dans la queue

```
# 20000 (défaut)
qmgr_message_active_limit = 20000

# Plus élevé pour gros volumes
qmgr_message_active_limit = 50000
```

SMTP destination concurrency limit

Nombre de connexions simultanées vers une même destination

```
# 20 (défaut)
smtp_destination_concurrency_limit = 20

# Plus agressif
smtp_destination_concurrency_limit = 50
```

✉ smtp_destination_rate_delay

Délai entre deux emails vers la même destination

```
# Pas de délai (défaut)
smtp_destination_rate_delay = 0s

# Limiter à 1 email par seconde
smtp_destination_rate_delay = 1s
```

Cas d'usage : Certains serveurs (Gmail, Yahoo) limitent le taux d'emails reçus. Si vous envoyez trop vite, ils vous throttle ou blacklist.

Paramètres de logging

maillog_file

Fichier de log personnalisé (Postfix 3.4+)

```
# Utiliser syslog (défaut)
maillog_file =  
  
# Fichier dédié
maillog_file = /var/log/postfix.log
```

🐛 debug_peer_list

Liste d'hôtes pour lesquels activer le debug

```
# Activer debug pour example.com  
debug_peer_list = example.com, 203.0.113.10
```

🔍 debug_peer_level

Niveau de debug

```
debug_peer_level = 2
```

Variables prédéfinies utiles

Postfix fournit des variables que vous pouvez utiliser :

```
$myhostname      # Nom du serveur  
$mydomain        # Domaine  
$myorigin        # Domaine d'origine  
$mail_name       # Postfix  
$mail_version    # 3.8.4
```

Exemple d'utilisation :

```
smtpd_banner = $myhostname ESMTP $mail_name  
# Devient : mail.example.com ESMTP Postfix
```



Configuration complète type

Voici un exemple de configuration complète et sécurisée pour 2025 :

```
# === IDENTITÉ ===
myhostname = mail.example.com
mydomain = example.com
myorigin = $mydomain
# === RÉSEAU ===
inet_interfaces = all
inet_protocols = ipv4
# === DESTINATIONS ===
mydestination = $myhostname, localhost.$mydomain, localhost, $mydomain
mynetworks = 127.0.0.0/8 [::ffff:127.0.0.0]/104 [::1]/128
relayhost =
# === STOCKAGE ===
home_mailbox = Maildir/
message_size_limit = 52428800
mailbox_size_limit = 0
```

```
# === SÉCURITÉ ===
smtpd_banner = $myhostname ESMTP
disable_vrfy_command = yes
smtpd_helo_required = yes

# === RESTRICTIONS ===
smtpd_recipient_restrictions =
    permit_mynetworks,
    permit_sasl_authenticated,
    reject_unauth_destination,
    reject_invalid_hostname,
    reject_non_fqdn_recipient,
    reject_unknown_recipient_domain
smtpd_helo_restrictions =
    permit_mynetworks,
    reject_invalid_helo_hostname,
    reject_non_fqdn_helo_hostname

smtpd_sender_restrictions =
    permit_mynetworks,
    permit_sasl_authenticated,
    reject_non_fqdn_sender,
    reject_unknown_sender_domain
```



```
# === TLS ===
smtpd_tls_cert_file = /etc/letsencrypt/live/mail.example.com/fullchain.pem
smtpd_tls_key_file = /etc/letsencrypt/live/mail.example.com/privkey.pem
smtpd_tls_security_level = may
smtpd_tls_protocols = >TLSv1.2
smtp_tls_security_level = may
smtp_tls_protocols = >TLSv1.2
```

(ne vous inquiétez pas nous le voyons dans le module TLS)



```
# === SASL ===
smtpd_sasl_auth_enable = yes
smtpd_sasl_type = dovecot
smtpd_sasl_path = private/auth
smtpd_sasl_security_options = noanonymous
```



```
# === PERFORMANCE ===
# je limite le nombre de processus Postfix simultanés à 100
default_process_limit = 100
# je limite le nombre de messages actifs dans la queue à 20000
qmgr_message_active_limit = 20000
# je limite la fréquence de traitement de la file d'attente à 300 secondes
queue_run_delay = 300s
# je limite le délai minimum avant de retester l'envoi à 300 secondes
minimal_backoff_time = 300s
# je limite le délai maximum entre deux tentatives à 4000 secondes
maximal_backoff_time = 4000s
# je limite la durée de vie de la file d'attente à 5 jours
maximal_queue_lifetime = 5d
# je limite la durée de vie de la file d'attente des messages rejetés à 5 jours
bounce_queue_lifetime = 5d
```



Commandes utiles pour main.cf

Voir toute la configuration

`postconf`

(Affiche TOUS les paramètres, même ceux par défaut)

Voir uniquement les paramètres modifiés

`postconf -n`

(Plus lisible !)

Voir un paramètre spécifique

```
postconf myhostname  
postconf message_size_limit
```

Modifier un paramètre

```
# Via commande  
sudo postconf -e "myhostname = mail.example.com"  
  
# Ou manuellement dans le fichier  
sudo nano /etc/postfix/main.cf
```

Recharger la configuration

```
# Vérifier d'abord  
sudo postfix check
```

```
# Recharger  
sudo systemctl reload postfix
```

Voir l'aide d'un paramètre

```
man 5 postconf
```

```
# Ou en ligne  
# http://www.postfix.org/postconf.5.html
```

Bonnes pratiques

💡 Commentez votre configuration

```
# === CONFIGURATION RÉSEAU ===
# Écoute sur toutes les interfaces pour accepter les emails de l'extérieur
inet_interfaces = all

# IPv4 uniquement car notre hébergeur ne fournit pas d'IPv6
inet_protocols = ipv4
```

📦 Gardez une sauvegarde

```
# Avant toute modification
sudo cp /etc/postfix/main.cf /etc/postfix/main.cf.backup

# Avec date
sudo cp /etc/postfix/main.cf /etc/postfix/main.cf.$(date +%Y%m%d)
```



✓ Testez toujours

```
# 1. Vérifier la syntaxe  
sudo postfix check  
  
# 2. Voir ce qui a changé  
sudo postconf -n > /tmp/new.conf  
diff /etc/postfix/main.cf.backup /tmp/new.conf  
  
# 3. Recharger  
sudo systemctl reload postfix  
  
# 4. Tester l'envoi  
echo "Test" | mail -s "Test" root
```

🔒 Sécurité avant performance

Préférez toujours une configuration sécurisée à une configuration ultra-performante mais risquée.

```
# ❌ Rapide mais dangereux
mynetworks = 0.0.0.0/0

# ✅ Plus lent mais sécurisé
mynetworks = 127.0.0.0/8
smtpd_sasl_auth_enable = yes
```

📊 Loggez suffisamment

```
# Trop peu de logs = impossible de débugger
smtpd_tls_loglevel = 0

# Bon compromis
smtpd_tls_loglevel = 1

# Pour debug uniquement
smtpd_tls_loglevel = 2
```

Exercices pratiques

🎯 Exercice 1 : Configuration de base

1. Modifiez `myhostname` pour correspondre à votre serveur
2. Configurez `mydomain` avec votre domaine
3. Vérifiez avec `postconf -n`
4. Testez l'envoi d'un email local

🎯 Exercice 2 : Restrictions

1. Ajoutez les restrictions recommandées
2. Testez d'envoyer un email sans authentification
3. Consultez les logs pour voir le rejet

🎯 Exercice 3 : Taille des messages

1. Limitez la taille des messages à 10 MB
2. Tentez d'envoyer un fichier de 15 MB
3. Observez l'erreur dans les logs

🎯 Exercice 4 : Bannière

1. Changez la bannière pour masquer la version de Postfix
2. Testez avec telnet :
3. Vérifiez que la bannière ne révèle plus d'informations

```
telnet mail.andromed.cloud 25
```

Points clés à retenir

💡 Configuration

Fichier : /etc/postfix/main.cf

- **Format :** paramètre = valeur
- **Vérification :** postfix check
- **Rechargement :** systemctl reload postfix

Paramètres essentiels : myhostname , mydomain , myorigin (Identité)

- inet_interfaces , mynetworks (Réseau)
- mydestination , relay_domains (Destinations) - Restrictions (Sécurité)

Commandes utiles : postconf (Voir la config)

- postconf -n (Voir les modifs uniquement)
- postconf -e (Modifier un paramètre)



Prochaine étape

Maintenant que vous maîtrisez le `main.cf`, nous allons apprendre les Alias et Domaines virtuels.

Module suivant : Alias et Domaines virtuels →



QCM - Module 4 : Configuration main.cf

Question 1

Quel paramètre définit les domaines pour lesquels Postfix accepte les emails localement ?

- A) myorigin
- B) mydestination
- C) relay_domains
- D) virtual_alias_domains

Question 2

Pour éviter un open relay, il faut ABSOLUMENT avoir :

- A) smtpd_tls_security_level = encrypt
- B) reject_unauth_destination dans les restrictions
- C) smtpd_helo_required = yes
- D) disable_vrfy_command = yes



Question 3

Quelle commande affiche UNIQUEMENT les paramètres modifiés ?

- A) postconf
- B) postconf -n
- C) postconf -d
- D) postfix check

Question 4

Quel paramètre permet de limiter la taille maximale d'un message accepté par Postfix ?

- A) queue_minfree
- B) message_size_limit
- C) bounce_queue_lifetime
- D) smtp_tls_loglevel



Question 5

Comment supprimer proprement une surcharge `main.cf` pour revenir à la valeur par défaut ?

- A) Supprimer la ligne à la main
- B) `postconf -# paramètre`
- C) `postconf -X paramètre`
- D) `postconf -d paramètre`

Réponses - Module 4

Question 1 : Réponse B - `mydestination` liste les domaines pour lesquels Postfix **livre localement** les emails (domaines finaux).

Question 2 : Réponse B - `reject_unauth_destination` **rejette** tout email vers un domaine non autorisé. Sans ça = **OPEN RELAY** !

Question 3 : Réponse B - `postconf -n` affiche **uniquement** les paramètres modifiés (non par défaut).

Question 4 : Réponse B - `message_size_limit` exprime la taille en octets d'un message accepté. Augmentez-le pour permettre les pièces jointes volumineuses ou réduisez-le pour limiter.

Question 5 : Réponse C - `postconf -X paramètre` enlève l'override dans `main.cf`. Postfix rebasculera alors sur la valeur par défaut (visible via `postconf -d`).

Exercice pratique - Module 4

🎯 Objectif

Maîtriser la configuration du main.cf

📋 Tâches (20 minutes)

1. Paramètres de base : Configurez ces paramètres essentiels :

```
sudo postconf -e "message_size_limit = 20971520" # 20 MB
sudo postconf -e "smtpd_banner = \$myhostname ESMTP"
sudo postconf -e "disable_vrfy_command = yes"
```

2. Restrictions anti-spam : Ajoutez des restrictions basiques :

```
sudo postconf -e "smtpd_recipient_restrictions = permit_mynetworks, reject_unauth_destination"
```

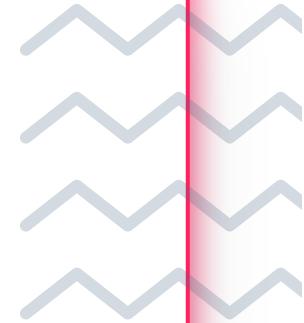


3. Vérification : Vérifiez votre config :

```
postfix check  
postconf -n | grep -E "(message_size|smtpd_banner|disable_vrfy|smtpd_recipient)"  
sudo systemctl reload postfix
```

4. Test : Testez l'envoi d'un email et vérifiez les logs





Alias et Tables Virtuelles

 Gérer des adresses email SANS créer d'utilisateurs système

Introduction

Situation réelle : Vous avez un serveur avec juste root et ubuntu .

Vous voulez gérer des centaines d'adresses email :

- johndoe@andromed.cloud
- janedoe@andromed.cloud
- contact@andromed.cloud
- etc.



Question : Faut-il créer 100 utilisateurs système ? 🤔

Réponse : NON ! On utilise les **domaines virtuels** ! 🎉*

Notre cas concret

Serveur OVH : Debian/Ubuntu avec :

- Utilisateur root
- Utilisateur ubuntu (créé par OVH)
- **C'est tout !**



Domaine : andromed.cloud

Emails à gérer :

- johndoe@andromed.cloud
- janedoe@andromed.cloud
- contact@andromed.cloud
- support@andromed.cloud



Tous ces emails SANS créer johndoe, janedoe comme utilisateurs système !



Les 3 façons de gérer les boîtes mail

Avant de commencer, comprenons bien les **3 méthodes** possibles avec Postfix.



1 Comptes système réels (utilisateurs Unix)

Principe : Chaque adresse mail = 1 utilisateur Unix

```
# User Unix : johndoe  
# Maildir : /home/johndoe/Maildir/  
johndoe@andromed.cloud ~ /home/johndoe/Maildir/
```

Dans ce cas il faut créer manuellement le dossier Maildir pour chaque utilisateur.

```
sudo mkdir -p /home/johndoe/Maildir/  
sudo chown -R johndoe:johndoe /home/johndoe/Maildir/  
sudo chmod -R 700 /home/johndoe/Maildir/
```

ou avec Dovecot :

```
sudo dovecadm exec dovecadm mailbox list -u johndoe
```

Avantages :

- Simple à mettre en place
- Postfix + Dovecot en mode "system users"

Inconvénients :

- 25 boîtes mail → 25 comptes Unix 😞
- Risques de sécurité (accès SSH à gérer)
- Pas scalable
- Gestion lourde

Verdict : **✗ OK pour 1-2 boîtes perso, pas pour un serveur professionnel**

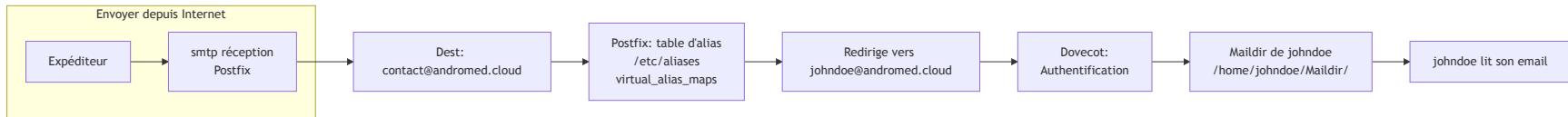


2 Aliases (redirections simples)

Principe : Pas de boîte mail, juste une redirection

```
# Dans /etc/aliases ou virtual_alias_maps
contact@andromed.cloud → johndoe@andromed.cloud
```

Un schéma d'explication :



Explication :

- Postfix reçoit un mail pour `contact@andromed.cloud`
- Il regarde dans la table d'alias : "Oh, cet alias redirige vers `johndoe@andromed.cloud`"
- Le mail est donc remis dans la boîte de `johndoe`.
- `johndoe` relève sa boîte (par IMAP/POP3)
- Il n'y a **pas** de Maildir pour `contact@andromed.cloud` : c'est juste une redirection.

Usage :

- Redirections internes (contact@ , support@)
- Regroupements
- Renvois externes

Important : ! Un alias n'est PAS une boîte mail

Il n'y a pas de Maildir séparé, c'est juste une règle de redirection.

Verdict : ✓ Utile, mais **complémentaire**, pas une solution complète



3 Virtual users (domaines virtuels)

Principe : Séparer les comptes mail des comptes Unix

```
# johndoe@andromed.cloud existe comme boîte mail  
# MAIS pas comme utilisateur Unix !  
johndoe@andromed.cloud ~ /var/mail/vhosts/andromed.cloud/johndoe/
```

Architecture :

- Adresses stockées dans Postfix (fichiers ou BDD)
- **1 seul utilisateur Unix** : vmail (UID 5000)
- vmail possède **TOUS** les Maildirs
- Dovecot + Postfix gèrent l'authentification

Avantages :

- Scalable (centaines d'adresses)
- Sécurisé (pas de comptes Unix multiples)
- Gestion centralisée
- Multi-domaines facile

Verdict : 🏆 C'est la méthode professionnelle !



Tableau récapitulatif

Méthode	Utilisateur Unix ?	Boîte mail réelle ?	Quand l'utiliser
User système	Oui	Oui	Petits serveurs perso
Alias	Non	Non (redirige)	contact@ , redirections
Virtual users	Non (1 seul vmail)	Oui	Serveurs pro 

🎯 Notre choix : Virtual users !

Dans ce module, nous allons utiliser la **3ème méthode** :

- Des centaines d'adresses SANS créer d'utilisateurs système
- Un seul user technique : vmail
- Architecture propre et scalable

Ce qu'on va faire

1. Vérifier les utilisateurs existants
2. Créer l'utilisateur technique vmail
3. Configurer les domaines virtuels
4. Créer des boîtes mail virtuelles
5. Tester avec Postfix + Dovecot



PARTIE 1 : État des lieux du serveur

Lister les utilisateurs actuels

```
# Voir tous les utilisateurs  
cat /etc/passwd
```



Sur un serveur OVH, vous verrez :

```
root:x:0:0:root:/root:/bin/bash
ubuntu:x:1000:1000:Ubuntu:/home/ubuntu:/bin/bash
...
(utilisateurs système : daemon, bin, sys, etc.)
```



Utilisateurs humains

```
# Lister uniquement les vrais utilisateurs (UID >= 1000)
awk -F: '$3 >= 1000 {print $1, "UID:", $3}' /etc/passwd
```

Résultat typique :

```
ubuntu UID: 1000
```



C'est parfait ! On ne veut PAS polluer le système avec des dizaines d'utilisateurs.



PARTIE 2 : Crée l'utilisateur technique vmail

Pourquoi vmail ?

vmail est un utilisateur technique qui va :

- Posséder TOUTES les boîtes mail virtuelles
- Permettre à Postfix et Dovecot d'y accéder
- Sécuriser les permissions



Création de vmail

```
# Créer le groupe vmail avec GID 5000  
sudo groupadd -g 5000 vmail
```



```
# Créer l'utilisateur vmail avec UID 5000
sudo useradd -g vmail -u 5000 vmail \
-d /var/mail/vhosts \
-s /sbin/nologin \
-c "Virtual Mail User"
```

Paramètres expliqués :

- **-g vmail** : Groupe principal
- **-u 5000** : UID fixe (important pour Dovecot)
- **-d /var/mail/vhosts** : Répertoire home
- **-s /sbin/nologin** : **PAS de login possible** (sécurité)
- **-c** : Commentaire

✓ Vérifier la création

id vmail

Résultat attendu :

```
uid=5000(vmail) gid=5000(vmail) groups=5000(vmail)
```

```
# Vérifier qu'on ne peut pas se connecter  
su - vmail
```

Résultat attendu :

```
This account is currently not available.
```

Parfait ! vmail existe mais personne ne peut s'y connecter. ✓

Créer la structure de réertoires

```
# Créer le répertoire principal  
sudo mkdir -p /var/mail/vhosts
```

```
# Créer le sous-répertoire pour notre domaine  
sudo mkdir -p /var/mail/vhosts/andromed.cloud
```

```
# Donner la propriété à vmail  
sudo chown -R vmail:vmail /var/mail/vhosts
```

```
# Permissions restrictives  
sudo chmod -R 770 /var/mail/vhosts
```



Vérifier

```
ls -ld /var/mail/vhosts  
ls -ld /var/mail/vhosts/andromed.cloud
```

Résultat attendu :

```
drwxrwx--- 3 vmail vmail 4096 Jan 15 14:00 /var/mail/vhosts  
drwxrwx--- 2 vmail vmail 4096 Jan 15 14:00 /var/mail/vhosts/andromed.cloud
```

PARTIE 3 : Configuration Postfix

Fichier des domaines virtuels

On va créer un fichier qui liste nos domaines gérés.

```
sudo nano /etc/postfix/virtual_domains
```

Dans : `/etc/postfix/virtual_domains`

Contenu :

```
# Domaines virtuels gérés par ce serveur  
andromed.cloud
```

Fichier des boîtes virtuelles

C'est ici qu'on déclare les adresses email et leur emplacement.

```
sudo nano /etc/postfix/vmailbox
```

Dans : `/etc/postfix/vmailbox`

Contenu :

```
# Format : adresse@domaine    chemin/relatif/
#
# Domaine : andromed.cloud
johndoe@andromed.cloud      andromed.cloud/johndoe/
janedoe@andromed.cloud      andromed.cloud/janedoe/
contact@andromed.cloud     andromed.cloud/contact/
support@andromed.cloud     andromed.cloud/support/
admin@andromed.cloud        andromed.cloud/admin/
```

Important : Le chemin est **relatif** à `/var/mail/vhosts/`

Donc `andromed.cloud/johndoe/` → `/var/mail/vhosts/andromed.cloud/johndoe/`

Compiler les fichiers

```
# Compiler vmailbox  
sudo postmap /etc/postfix/vmailbox
```

postmap est un outil de postfix qui compile les fichiers de configuration.

Vérifier :

```
ls -l /etc/postfix/vmailbox.db
```



Configuration dans main.cf

```
sudo nano /etc/postfix/main.cf
```

Ajouter à la fin :

```
# =====#
# DOMAINES VIRTUELS (Virtual Mailbox)
# =====#
# Domaines virtuels avec vraies boîtes mail
virtual_mailbox_domains = andromed.cloud

# Mapping adresse → chemin mailbox
virtual_mailbox_maps = hash:/etc/postfix/vmailbox

# Répertoire de base pour les mailbox
virtual_mailbox_base = /var/mail/vhosts

# UID et GID de vmail
virtual_uid_maps = static:5000
virtual_gid_maps = static:5000

# Transport pour les emails (important)
virtual_transport = virtual
```



N'oubliez pas !

N'oubliez pas de retirer home_mailbox dans main.cf

```
# N'oubliez pas de retirer home_mailbox dans main.cf
#home_mailbox = Maildir/
```

Car on va utiliser le transport virtual pour les emails.

N'oubliez pas d'enlever \$mydomain dans mydestination du fichier main.cf

Si non vous aurez :

2025-11-17T11:56:08.083137+00:00 vps-02de336b postfix/trivial-rewrite[74147]: warning: do not list domain andromed.cloud in BOTH mydestination and virtual_mailbox_domains



Bonus de sécurité :

```
# Taille maximum d'une boîte mail (100 Mo)
virtual_mailbox_limit = 104857600

# UID minimum (sécurité)
virtual_minimum_uid = 5000
```

Explication :

- `virtual_mailbox_domains` : Domaines avec boîtes réelles
- `virtual_mailbox_maps` : Fichier de mapping
- `virtual_mailbox_base` : Racine des boîtes
- `virtual_uid_maps` / `virtual_gid_maps` : Propriétaire (vmail)
- `virtual_mailbox_limit` : Quota par boîte
- `virtual_transport` : Transport pour les emails, si non précisé par défaut : local

Vérifier la configuration

`postfix check`

Si pas d'erreur, c'est bon ! 

Recharger Postfix

`sudo systemctl reload postfix`

`sudo systemctl status postfix`



PARTIE 4 : Tests avec Postfix seul

Envoyer un email de test

```
echo "Ceci est un test pour johndoe" | \  
mail -s "Test Virtual Mailbox" johndoe@andromed.cloud
```

Suivre les logs

```
sudo tail -f /var/log/mail.log
```

Vous devriez voir :

```
postfix/virtual[1234]: delivered to mailbox
to=<johndoe@andromed.cloud>,
relay=virtual,
status=sent (delivered to maildir)
```



Vérifier que le fichier existe

```
sudo ls -la /var/mail/vhosts/andromed.cloud/john Doe/
```



Vous devriez voir :

```
drwx----- 5 vmail vmail 4096 Jan 15 14:30 .
drwxrwx--- 3 vmail vmail 4096 Jan 15 14:00 ..
drwx----- 2 vmail vmail 4096 Jan 15 14:30 cur
drwx----- 2 vmail vmail 4096 Jan 15 14:30 new
drwx----- 2 vmail vmail 4096 Jan 15 14:30 tmp
```

Format Maildir : 3 dossiers

- new/ : Nouveaux emails
- cur/ : Emails lus
- tmp/ : Temporaire



Voir l'email reçu

```
sudo ls /var/mail/vhosts/andromed.cloud/johndoe/new/
```

puis : **cat** pour voir le contenu de l'email

```
sudo cat /var/mail/vhosts/andromed.cloud/johndoe/new/*
```

Vous verrez le contenu brut de l'email ! 



PARTIE 5 : Alias et redirections

Rediriger plusieurs adresses vers une boîte

Besoin : contact@ et info@ vont vers la même boîte



Fichier virtual_alias

```
sudo nano /etc/postfix/virtual_alias
```

[Revenir au sommaire](#)

Contenu :

```
# Redirections d'alias vers boîtes virtuelles
#
# Domaine : andromed.cloud
info@andromed.cloud      john Doe@andromed.cloud
webmaster@andromed.cloud admin@andromed.cloud
```



Compiler

```
sudo postmap /etc/postfix/virtual_alias
```

Configuration main.cf

```
sudo nano /etc/postfix/main.cf
```

Ajouter à la fin :

```
# Alias virtuels (redirections)
virtual_alias_maps = hash:/etc/postfix/virtual_alias
```

```
sudo systemctl reload postfix
```

Tester

```
echo "Test alias" | mail -s "Info" info@andromed.cloud
```



```
# Doit arriver dans la boîte de johndoe  
sudo ls /var/mail/vhosts/andromed.cloud/johndoe/new/
```



Redirection vers email externe

Besoin : facturation@andromed.cloud → votre comptable externe



Éditer virtual_alias

```
sudo nano /etc/postfix/virtual_alias
```

Ajouter :

```
# Redirection externe  
facturation@andromed.cloud    comptable@cabinet-expert.fr  
  
sudo postmap /etc/postfix/virtual_alias  
sudo systemctl reload postfix
```

 **Tester**

```
echo "Facture" | mail -s "Facture #123" facturation@andromed.cloud
```

Le comptable doit recevoir l'email ! 



Redirection multiple

Besoin : support@ envoie à plusieurs personnes

```
sudo nano /etc/postfix/virtual_alias
```

Ajouter :

```
# Support vers plusieurs boîtes  
support@andromed.cloud john Doe <johndoe@andromed.cloud>, Jane Doe <janedoe@andromed.cloud>, Admin <admin@andromed.cloud>  
  
sudo postmap /etc/postfix/virtual_alias  
sudo systemctl reload postfix
```



```
echo "Question support" | mail -s "Ticket #456" support@andromed.cloud
```

[Revenir au sommaire](#)

Les 3 boîtes doivent recevoir l'email :

```
sudo ls /var/mail/vhosts/andromed.cloud/johndoe/new/  
sudo ls /var/mail/vhosts/andromed.cloud/janedoe/new/  
sudo ls /var/mail/vhosts/andromed.cloud/admin/new/
```

PARTIE 6 : Catch-all (bonus)

Attraper toutes les adresses non définies

Besoin : Toute adresse `*@andromed.cloud` non définie va chez admin

Comme je vous le précise plus tard, c'est du bonus, nous n'allons pas nous attarder sur cette partie car elle est sujette à recevoir beaucoup de spam.

Éditer virtual_alias

```
sudo nano /etc/postfix/virtual_alias
```

[Revenir au sommaire](#)



Ajouter À LA FIN :

```
# Catch-all pour andromed.cloud (DOIT être à la fin)
@andromed.cloud          admin@andromed.cloud
```



⚠️ Important : Les règles spécifiques AVANT le catch-all !

```
# ✅ BON ORDRE
contact@andromed.cloud      john doe@andromed.cloud
info@andromed.cloud          john doe@andromed.cloud
@andromed.cloud               admin@andromed.cloud

# ❌ MAUVAIS ORDRE (catch-all écrase tout)
@andromed.cloud              admin@andromed.cloud
contact@andromed.cloud       john doe@andromed.cloud  # jamais atteint !
```

```
sudo postmap /etc/postfix/virtual_alias  
sudo systemctl reload postfix
```



Tester

```
echo "Test catch-all" | mail -s "Test" nimportequoi@andromed.cloud
```



```
# Doit arriver chez admin  
sudo ls /var/mail/vhosts/andromed.cloud/admin/new/
```



⚠ Danger : Les catch-all attirent BEAUCOUP de spam !

```
randomspam@andromed.cloud → admin  
phishing@andromed.cloud → admin  
test@andromed.cloud → admin  
...
```

Recommandation : Évitez les catch-all ou utilisez un bon anti-spam (Rspamd)



PARTIE 7 : Gérer plusieurs domaines

Ajouter un deuxième domaine

Nouveau domaine : ascent.cloud

Même serveur, domaine différent.



Éditer virtual_domains

```
sudo nano /etc/postfix/virtual_domains
```



Ajouter :

```
# Domaines virtuels gérés  
andromed.cloud  
ascent.cloud
```



Créer le répertoire

```
sudo mkdir -p /var/mail/vhosts/ascent.cloud  
sudo chown vmail:vmail /var/mail/vhosts/ascent.cloud  
sudo chmod 770 /var/mail/vhosts/ascent.cloud
```



Ajouter les boîtes dans vmailbox

```
sudo nano /etc/postfix/vmailbox
```



Ajouter :

```
# Domaine : ascent.cloud
contact@ascent.cloud      ascent.cloud/contact/
admin@ascent.cloud        ascent.cloud/admin/
```



```
sudo postmap /etc/postfix/vmailbox
```



Mettre à jour main.cf

```
sudo nano /etc/postfix/main.cf
```

Modifier :

```
# Domaines virtuels avec vraies boîtes mail  
virtual_mailbox_domains = andromed.cloud, ascent.cloud  
sudo systemctl reload postfix
```



```
echo "Test Ascent" | mail -s "Test" contact@ascent.cloud  
sudo ls /var/mail/vhosts/ascent.cloud/contact/new/
```

PARTIE 8 : Alias système (root, postmaster)

Gérer les alias système

Les alias comme `root@` , `postmaster@` doivent être redirigés.



Fichier /etc/aliases

`sudo nano /etc/aliases`

Modifier/Ajouter :

```
# Redirections des comptes système
postmaster: root
hostmaster: root
webmaster: root
abuse: root
security: root
noc: root
```

```
# Rediriger root vers une vraie boîte  
root: admin@andromed.cloud
```

Puis :

```
sudo newaliases
```



```
echo "Test postmaster" | mail -s "Test" postmaster
```

```
# Doit arriver dans admin@andromed.cloud  
sudo ls /var/mail/vhosts/andromed.cloud/admin/new/
```



PARTIE 9 : Intégration avec Dovecot

(Veuillez lire avant le module Dovecot)

[Lire le module Dovecot](#)

 **Point important à comprendre :**

Jusqu'ici, avec Postfix seul, on peut **RECEVOIR** des emails dans les boîtes virtuelles.

Mais les utilisateurs ne peuvent **PAS encore les lire !**



Pourquoi ?

- Postfix = Facteur qui dépose le courrier ✓
- **Pas besoin de mot de passe** pour recevoir des emails
- Les emails arrivent automatiquement dans /var/mail/vhosts/

Maintenant avec Dovecot :

- Dovecot = La clé de la boîte aux lettres 🔑
- **Besoin d'un mot de passe** pour ouvrir la boîte
- Permet la lecture via IMAP/POP3
- Fournit l'authentification SMTP pour l'envoi

Passons à la configuration !

Configuration Dovecot

Fichier 10-mail.conf

```
sudo nano /etc/dovecot/conf.d/10-mail.conf
```

Configurer :

```
# Location des boîtes mail
mail_location = maildir:/var/mail/vhosts/%d/%n

# Utilisateur et groupe
mail_uid = vmail
mail_gid = vmail
first_valid_uid = 5000
first_valid_gid = 5000
```

Variables Dovecot :

- %d = domaine (andromed.cloud)
- %n = utilisateur local (johndoe)
- %u = adresse complète (johndoe@andromed.cloud)

Fichier 10-auth.conf

```
sudo nano /etc/dovecot/conf.d/10-auth.conf
# Mécanismes d'authentification
auth_mechanisms = plain login
# Désactiver l'auth en clair (sauf SSL)
disable_plaintext_auth = yes
```



Créer le fichier d'utilisateurs virtuels

```
sudo nano /etc/dovecot/users
```

Contenu :

```
# Format : utilisateur@domaine:{PLAIN}motdepasse
johndoe@andromed.cloud:{PLAIN}MotDePasse123!
janedoe@andromed.cloud:{PLAIN}MotDePasse456!
contact@andromed.cloud:{PLAIN}MotDePasse789!
admin@andromed.cloud:{PLAIN}AdminPass123!
```

⚠ Important : En production, utilisez des hash (SHA512-CRYPT) !

```
# Générer un hash  
doveadm pw -s SHA512-CRYPT
```



Configuration de l'authentification

```
sudo nano /etc/dovecot/conf.d/auth-passwdfile.conf.ext
```



Créer/Modifier :

```
passdb {  
    driver = passwd-file  
    args = scheme=PLAIN username_format=%u /etc/dovecot/users  
}  
  
userdb {  
    driver = static  
    args = uid=vmail gid=vmail home=/var/mail/vhosts/%d/%n  
}
```



Activer cette authentification

```
sudo nano /etc/dovecot/conf.d/10-auth.conf
```



Commenter les auth par défaut et ajouter :

```
# Désactiver
#!include auth-system.conf.ext

# Activer
!include auth-passwdfile.conf.ext
```

Vous pouvez aussi tout faire dans le auth-system.conf d'ailleurs, libre à vous en terme de propreté ou non.

Permissions

```
sudo chmod 640 /etc/dovecot/users
sudo chown root:dovecot /etc/dovecot/users
```



Redémarrer Dovecot

```
sudo systemctl restart dovecot  
sudo systemctl status dovecot
```



Tester l'authentification

```
doveadm auth test johndoe@andromed.cloud MotDePasse123!
```

Résultat attendu :

```
passdb: johndoe@andromed.cloud auth succeeded
userdb: johndoe@andromed.cloud
  home   : /var/mail/vhosts/andromed.cloud/johndoe
  uid    : 5000
  gid    : 5000
```

Tester avec un client email

Paramètres de configuration

Serveur IMAP :

- Serveur : mail.andromed.cloud
- Port : 993 (IMAPS)
- SSL/TLS : Activé
- Utilisateur : johndoe@andromed.cloud
- Mot de passe : MotDePasse123!



Serveur SMTP :

- Serveur : mail.andromed.cloud
- Port : 587 (Submission)
- STARTTLS : Activé
- Authentification : Oui
- Utilisateur : johndoe@andromed.cloud
- Mot de passe : MotDePasse123!



PARTIE 10 : Commandes utiles

Lister les boîtes mail

```
# Toutes les boîtes
sudo find /var/mail/vhosts -name "new" -type d
```

Compter les emails par boîte

```
# Emails de johndoe
sudo ls /var/mail/vhosts/andromed.cloud/johndoe/new/ | wc -l
```

Lire un email

```
# Lister les emails
sudo ls /var/mail/vhosts/andromed.cloud/johndoe/new/
```

```
# Lire un email spécifique
sudo cat /var/mail/vhosts/andromed.cloud/johndoe/new/1234567890.V1I2M3.mail
```

Vérifier une adresse

```
# Vérifier si l'adresse existe dans vmailbox  
postmap -q johndoe@andromed.cloud /etc/postfix/vmailbox
```

Résultat :

```
andromed.cloud/johndoe/
```



Vérifier un alias

```
# Vérifier une redirection  
postmap -q info@andromed.cloud /etc/postfix/virtual_alias
```

Résultat :

```
johndoe@andromed.cloud
```



Purger une boîte

```
# Supprimer tous les emails de johndoe  
sudo rm -rf /var/mail/vhosts/andromed.cloud/johndoe/new/*  
sudo rm -rf /var/mail/vhosts/andromed.cloud/johndoe/cur/*
```

Taille des boîtes

```
# Taille de toutes les boîtes  
sudo du -sh /var/mail/vhosts/andromed.cloud/*/
```



```
# Taille d'une boîte spécifique  
sudo du -sh /var/mail/vhosts/andromed.cloud/johndoe/
```

PARTIE 11 : Scripts d'administration

Script : Créer une nouvelle adresse

```
sudo nano /usr/local/bin/add-virtual-mailbox.sh
```



Contenu :

```
#!/bin/bash
# add-virtual-mailbox.sh

if [ "$#" -ne 2 ]; then
    echo "Usage: $0 email@domain password"
    exit 1
fi

EMAIL=$1
PASSWORD=$2
DOMAIN=$(echo $EMAIL | cut -d@ -f2)
USER=$(echo $EMAIL | cut -d@ -f1)
```

```
# Ajouter à vmailbox
echo "$EMAIL    $DOMAIN/$USER/" | sudo tee -a /etc/postfix/vmailbox
sudo postmap /etc/postfix/vmailbox

# Ajouter à Dovecot users
echo "$EMAIL:{PLAIN}$PASSWORD" | sudo tee -a /etc/dovecot/users
```

```
# Créer le répertoire
sudo mkdir -p /var/mail/vhosts/$DOMAIN/$USER/{new,cur,tmp}
sudo chown -R vmail:vmail /var/mail/vhosts/$DOMAIN/$USER
sudo chmod -R 700 /var/mail/vhosts/$DOMAIN/$USER

# Recharger
sudo systemctl reload postfix
sudo systemctl reload dovecot

echo "✅ Boîte mail $EMAIL créée !"
```



```
# Rendre exécutables  
sudo chmod +x /usr/local/bin/add-virtual-mailbox.sh
```



💡 Utiliser le script

```
sudo /usr/local/bin/add-virtual-mailbox.sh test@andromed.cloud Password123
```

Script : Supprimer une adresse

```
sudo nano /usr/local/bin/remove-virtual-mailbox.sh
```



Contenu :

```
#!/bin/bash
# remove-virtual-mailbox.sh

if [ "$#" -ne 1 ]; then
    echo "Usage: $0 email@domain"
    exit 1
fi

EMAIL=$1
DOMAIN=$(echo $EMAIL | cut -d@ -f2)
USER=$(echo $EMAIL | cut -d@ -f1)
```

```
# Supprimer de vmailbox
sudo sed -i "/^$EMAIL[:space:]/d" /etc/postfix/vmailbox
sudo postmap /etc/postfix/vmailbox

# Supprimer de Dovecot users
sudo sed -i "/^$EMAIL:/d" /etc/dovecot/users
```

```
# Supprimer le répertoire  
sudo rm -rf /var/mail/vhosts/$DOMAIN/$USER  
  
# Recharger  
sudo systemctl reload postfix  
sudo systemctl reload dovecot  
  
echo "☒ Boîte mail $EMAIL supprimée!"
```



```
sudo chmod +x /usr/local/bin/remove-virtual-mailbox.sh
```



Script : Lister toutes les adresses

```
sudo nano /usr/local/bin/list-virtual-mailboxes.sh
```



Contenu :

```
#!/bin/bash
# list-virtual-mailboxes.sh

echo "=====
echo " BOÎTES MAIL VIRTUELLES"
echo "=====
echo ""

grep -v '^#' /etc/postfix/vmailbox | grep -v '^$' | while read line; do
    email=$(echo $line | awk '{print $1}')
    path=$(echo $line | awk '{print $2}')
    count=$(sudo ls /var/mail/vhosts/$path/new 2>/dev/null | wc -l)
    size=$(sudo du -sh /var/mail/vhosts/$path 2>/dev/null | awk '{print $1}')
    printf "%-40s [%3d emails] [%s]\n" "$email" "$count" "$size"
done

echo ""
echo "=====
```



```
sudo chmod +x /usr/local/bin/list-virtual-mailboxes.sh
```



Utiliser le script

```
sudo /usr/local/bin/list-virtual-mailboxes.sh
```

Résultat :

```
=====
BOÎTES MAIL VIRTUELLES
=====

johndoe@andromed.cloud      [ 5 emails] [128K]
janedoe@andromed.cloud      [ 2 emails] [64K]
contact@andromed.cloud      [ 12 emails] [256K]
admin@andromed.cloud        [ 8 emails] [192K]

=====
```

PARTIE 12 : Troubleshooting

✗ Problème : Email non délivré

Vérifier les logs

```
sudo tail -50 /var/log/mail.log | grep ERROR
```

Erreur courante 1

User unknown in virtual mailbox table

Cause : L'adresse n'existe pas dans vmailbox

Solution :

```
# Vérifier  
postmap -q johndoe@andromed.cloud /etc/postfix/vmailbox  
  
# Si vide, ajouter  
sudo nano /etc/postfix/vmailbox  
# Ajouter la ligne  
sudo postmap /etc/postfix/vmailbox
```

Erreur courante 2

Unable to create file /var/mail/vhosts/.../new/: Permission denied

Cause : Problème de permissions



Solution :

```
# Vérifier les permissions  
ls -ld /var/mail/vhosts/  
ls -ld /var/mail/vhosts/andromed.cloud/
```

```
# Corriger  
sudo chown -R vmail:vmail /var/mail/vhosts  
sudo chmod -R 770 /var/mail/vhosts
```

Erreur courante 3

```
fatal: parameter virtual_uid_maps: unknown user name: vmail
```

Cause : L'utilisateur vmail n'existe pas



Solution :

```
# Vérifier  
id vmail  
  
# Si erreur, recréer  
sudo groupadd -g 5000 vmail  
sudo useradd -g vmail -u 5000 vmail -d /var/mail/vhosts -s /sbin/nologin
```

✖ Problème : Authentification Dovecot échoue

Vérifier les logs

```
sudo tail -50 /var/log/dovecot/dovecot.log | grep auth
```



Tester l'authentification

```
doveadm auth test johndoe@andromed.cloud MotDePasse123!
```



Si erreur :

auth failed



Vérifier le fichier users :

```
sudo cat /etc/dovecot/users | grep johndoe
```



Format correct :

johndoe@andromed.cloud:{PLAIN}MotDePasse123!

Permissions :

```
sudo chmod 640 /etc/dovecot/users
sudo chown root:dovecot /etc/dovecot/users
```



🔍 Mode debug Postfix

```
# Activer  
sudo postconf -e "debug_peer_list=127.0.0.1"  
sudo postconf -e "debug_peer_level=3"  
sudo systemctl reload postfix
```

```
# Tester  
echo "Debug test" | mail -s "Test" johndoe@andromed.cloud
```



```
# Voir les logs détaillés  
sudo tail -f /var/log/mail.log
```



```
# Désactiver
sudo postconf -e "debug_peer_list="
sudo postconf -e "debug_peer_level=0"
sudo systemctl reload postfix
```



PARTIE 13 : Exercices pratiques

🎯 Exercice 1 : Première boîte virtuelle

Objectif : Créer votre première adresse complète

Tâches :

1. Créer alice@andromed.cloud dans vmailbox
2. Ajouter le mot de passe dans Dovecot users
3. Compiler et recharger
4. Envoyer un email de test
5. Vérifier que l'email est arrivé
6. Tester l'authentification avec doveadm

✓ Solution Exercice 1

```
# 1. Ajouter dans vmailbox  
sudo nano /etc/postfix/vmailbox
```

```
# Ajouter  
alice@andromed.cloud      andromed.cloud/alice/
```

```
# 2. Compiler  
sudo postmap /etc/postfix/vmailbox
```



```
# 3. Créer le répertoire  
sudo mkdir -p /var/mail/vhosts/andromed.cloud/alice/{new,cur,tmp}  
sudo chown -R vmail:vmail /var/mail/vhosts/andromed.cloud/alice  
sudo chmod -R 700 /var/mail/vhosts/andromed.cloud/alice
```



```
# 4. Ajouter dans Dovecot
sudo nano /etc/dovecot/users

# Ajouter
alice@andromed.cloud:{PLAIN}AlicePass123!
```



```
# 5. Recharger  
sudo systemctl reload postfix  
sudo systemctl reload dovecot
```



```
# 6. Tester envoi  
echo "Bonjour Alice" | mail -s "Bienvenue" alice@andromed.cloud
```



7. Vérifier

```
sudo ls /var/mail/vhosts/andromed.cloud/alice/new/  
sudo cat /var/mail/vhosts/andromed.cloud/alice/new/*
```

[Revenir au sommaire](#)

```
# 8. Tester auth  
doveadm auth test alice@andromed.cloud AlicePass123!
```



🎯 Exercice 2 : Alias et redirections

Objectif : Créer des redirections complexes

Tâches :

1. Créer info@andromed.cloud qui redirige vers alice
2. Créer team@andromed.cloud vers alice + johndoe
3. Créer billing@andromed.cloud vers votre email externe
4. Tester les 3 redirections

✓ Solution Exercice 2

```
# 1. Éditer virtual_alias  
sudo nano /etc/postfix/virtual_alias
```

Ajouter :

```
# Exercice 2  
info@andromed.cloud      alice@andromed.cloud  
team@andromed.cloud      alice@andromed.cloud, johndoe@andromed.cloud  
billing@andromed.cloud   votre@email.com
```

```
# 2. Compiler et recharger  
sudo postmap /etc/postfix/virtual_alias  
sudo systemctl reload postfix
```



```
# 3. Tests
echo "Test info" | mail -s "Info" info@andromed.cloud
echo "Test team" | mail -s "Team" team@andromed.cloud
echo "Test billing" | mail -s "Billing" billing@andromed.cloud
```



```
# 4. Vérifier  
sudo ls /var/mail/vhosts/andromed.cloud/alice/new/  
sudo ls /var/mail/vhosts/andromed.cloud/johndoe/new/  
# Vérifier votre boîte externe
```

🎯 Exercice 3 : Deuxième domaine

Objectif : Gérer un second domaine

Tâches :

1. Ajouter le domaine exemple.local
2. Créer contact@example.local
3. Créer admin@example.local
4. Configurer un catch-all vers admin
5. Tester

✓ Solution Exercice 3

```
# 1. Créer le répertoire  
sudo mkdir -p /var/mail/vhosts/exemple.local  
sudo chown vmail:vmail /var/mail/vhosts/exemple.local  
sudo chmod 770 /var/mail/vhosts/exemple.local
```

```
# 2. Ajouter dans vmailbox
sudo nano /etc/postfix/vmailbox

# Ajouter
contact@example.local      exemple.local/contact/
admin@example.local        exemple.local/admin/
```



```
sudo postmap /etc/postfix/vmailbox
```



```
# 3. Créer les répertoires
sudo mkdir -p /var/mail/vhosts/exemple.local/contact/{new,cur,tmp}
sudo mkdir -p /var/mail/vhosts/exemple.local/admin/{new,cur,tmp}
sudo chown -R vmail:vmail /var/mail/vhosts/exemple.local
sudo chmod -R 700 /var/mail/vhosts/exemple.local/contact
sudo chmod -R 700 /var/mail/vhosts/exemple.local/admin
```

```
# 4. Ajouter les mots de passe Dovecot
sudo nano /etc/dovecot/users

# Ajouter
contact@example.local:{PLAIN}ContactPass!
admin@example.local:{PLAIN}AdminPass!
```



```
# 5. Configurer catch-all  
sudo nano /etc/postfix/virtual_alias
```

```
# Ajouter A LA FIN  
@exemple.local      admin@example.local
```



```
sudo postmap /etc/postfix/virtual_alias
```



```
# 6. Mettre à jour main.cf  
sudo nano /etc/postfix/main.cf  
  
# Modifier  
virtual_mailbox_domains = andromed.cloud, exemple.local
```



```
# 7. Recharger  
sudo systemctl reload postfix  
sudo systemctl reload dovecot
```



```
# 8. Tests
echo "Test 1" | mail -s "Contact" contact@exemple.local
echo "Test 2" | mail -s "Admin" admin@exemple.local
echo "Test 3" | mail -s "Random" random@exemple.local
```



9. Vérifier

```
sudo ls /var/mail/vhosts/exemple.local/contact/new/  
sudo ls /var/mail/vhosts/exemple.local/admin/new/
```



🎯 Exercice 4 : Script automatisé

Objectif : Utiliser les scripts d'administration



Tâches :

1. Utiliser le script pour créer bob@andromed.cloud
2. Lister toutes les boîtes avec le script de listing
3. Envoyer un email à bob
4. Vérifier avec le script que bob a 1 email
5. Supprimer bob avec le script

✓ Solution Exercice 4

```
# 1. Créer bob  
sudo /usr/local/bin/add-virtual-mailbox.sh bob@andromed.cloud BobPass123
```



```
# 2. Lister
sudo /usr/local/bin/list-virtual-mailboxes.sh

# 3. Envoyer email
echo "Salut Bob" | mail -s "Test" bob@andromed.cloud

# 4. Relister (bob doit avoir 1 email)
sudo /usr/local/bin/list-virtual-mailboxes.sh

# 5. Supprimer
sudo /usr/local/bin/remove-virtual-mailbox.sh bob@andromed.cloud

# 6. Vérifier que bob est supprimé
sudo /usr/local/bin/list-virtual-mailboxes.sh
```

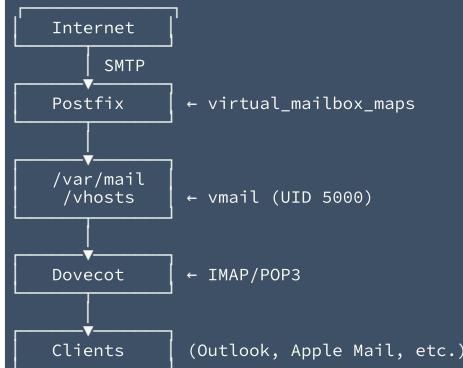
PARTIE 14 : Récapitulatif

🎯 **Ce que vous savez maintenant**

✓ **Vous savez gérer des centaines d'emails SANS utilisateurs système**

```
johndoe@andromed.cloud → /var/mail/vhosts/andromed.cloud/johndoe/  
janedoe@andromed.cloud → /var/mail/vhosts/andromed.cloud/janedoe/  
...
```

✓ Architecture complète



Points clés

1. Un seul utilisateur technique : vmail

- UID/GID 5000
- Pas de login possible
- Possède toutes les boîtes

2. Fichiers Postfix

```
/etc/postfix/vmailbox      # Boites virtuelles  
/etc/postfix/virtual_alias # Redirections  
/etc/postfix/virtual_domains # Domaines gérés
```



3. Fichiers Dovecot

```
/etc/dovecot/users      # Utilisateurs + mots de passe  
/etc/dovecot/conf.d/10-mail.conf  # Configuration maildir  
/etc/dovecot/conf.d/10-auth.conf  # Configuration auth
```

4. Structure des répertoires

```
/var/mail/vhosts/  
└── andromed.cloud/  
    ├── johndoe/  
    │   ├── new/      (nouveaux emails)  
    │   ├── cur/     (emails lus)  
    │   └── tmp/     (temporaire)  
    ├── janedoe/  
    ├── contact/  
    └── exemple.local/  
        ├── contact/  
        └── admin/
```



5. Commandes essentielles

```
# Ajouter une boîte  
sudo nano /etc/postfix/vmailbox  
sudo postmap /etc/postfix/vmailbox  
  
# Ajouter un alias  
sudo nano /etc/postfix/virtual_alias  
sudo postmap /etc/postfix/virtual_alias
```



```
# Vérifier  
postmap -q EMAIL /etc/postfix/vmailbox  
postmap -q EMAIL /etc/postfix/virtual_alias  
  
# Recharger  
sudo systemctl reload postfix  
sudo systemctl reload dovecot
```

6. Ordre de résolution

1. virtual_alias_maps (redirections)
2. virtual_mailbox_maps (boîtes réelles)
3. Livraison

Sécurité

Bonnes pratiques

- Utilisateur vmail sans login
- Permissions 770 sur /var/mail/vhosts
- Hash des mots de passe (SHA512-CRYPT)
- SSL/TLS obligatoire
- Pas de catch-all (ou avec anti-spam)

⚠️ À éviter

- ✗ Créer des utilisateurs système pour chaque email
- ✗ Permissions trop permissives (777)
- ✗ Mots de passe en clair visibles
- ✗ Catch-all sans protection
- ✗ Pas de quota (risque de saturation disque)



Pour aller plus loin

Base de données MySQL/PostgreSQL

Au lieu de fichiers texte, utiliser une BDD :

```
virtual_mailbox_maps = mysql:/etc/postfix/mysql-virtual-mailbox.cf  
virtual_alias_maps = mysql:/etc/postfix/mysql-virtual-alias.cf
```

Interface web de gestion

- **Postfixadmin** : Gérer les domaines/boîtes via web
- **Roundcube** : Webmail
- **iRedMail** : Solution complète

Quotas avancés

```
# Dans Dovecot
plugin {
    quota = maildir:User quota
    quota_rule = *:storage=1GB
}
```

[Revenir au sommaire](#)



Félicitations !

Vous savez maintenant :

- Gérer des centaines d'adresses email sans polluer le système
- Créer des boîtes virtuelles avec Postfix + Dovecot
- Faire des redirections simples et complexes
- Gérer plusieurs domaines sur un serveur
- Automatiser avec des scripts
- Déboguer les problèmes courants



Prochaine étape

Maintenant que vous maîtrisez les domaines virtuels, passons à la **protection anti-spam !** 🛡

Module suivant : Protection anti-spam →

QCM - Module 6 : Alias et tables virtuelles

Question 1

Après avoir modifié `/etc/aliases` , quelle commande faut-il exécuter ?

- A) `postfix reload`
- B) `newaliases`
- C) `systemctl restart postfix`
- D) Aucune, c'est automatique

Question 2

Quelle est la différence entre alias et virtual ?

- A) Il n'y a pas de différence
- B) Alias = domaines locaux, Virtual = domaines non locaux
- C) Virtual = plus récent que Alias
- D) Alias = plus sécurisé



Question 3

Comment créer un catch-all pour un domaine virtuel ?

- A) *@example.com admin@example.com
- B) @example.com admin@example.com
- C) catchall@example.com admin@example.com
- D) example.com admin@example.com

Question 4

Quel type de table est le plus couramment utilisé pour `virtual_alias_maps` sur une installation Postfix standard ?

- A) `btree:/...`
- B) `hash:/...`
- C) `texthash:/...`
- D) `ldap:/...`

Question 5

Quelle commande permet de tester la résolution d'un alias virtuel spécifique ?

- A) `postconf virtual_alias_maps`
- B) `postalias -q`
- C) `postmap -q user@example.com /etc/postfix/virtual`
- D) `postqueue -q user@example.com`

Réponses - Module 6

Question 1 : Réponse B - newaliases (ou postalias /etc/aliases) compile le fichier en base de données binaire .db que Postfix utilise.

Question 2 : Réponse B - Alias fonctionne pour les domaines dans mydestination (locaux). **Virtual** pour les domaines NON locaux (dans virtual_alias_domains).

Question 3 : Réponse B - @example.com admin@example.com capture TOUS les emails pour example.com. Attention à le placer EN DERNIER dans le fichier !

Question 4 : Réponse B - Le backend **hash** (fichiers .db générés par postmap) est disponible partout et performant pour quelques centaines d'entrées.

Question 5 : Réponse C - postmap -q adresse table interroge la table exactement comme Postfix le ferait, pratique pour valider une entrée sans envoyer d'email.

Exercice pratique - Module 6

🎯 Objectif

Configurer des alias et domaines virtuels

📋 Tâches (20 minutes)

1. Créer des alias locaux :

```
sudo nano /etc/aliases
# Ajouter :
# postmaster: $USER
# webmaster: $USER
# abuse: $USER

sudo newaliases
```

2. Tester les alias :

```
echo "Test alias" | mail -s "Test" postmaster  
ls ~/Maildir/new/
```

3. Configurer un domaine virtuel :

```
# Dans main.cf
sudo postconf -e "virtual_alias_domains = test.local"
sudo postconf -e "virtual_alias_maps = hash:/etc/postfix/virtual"

# Créer /etc/postfix/virtual
echo "admin@test.local $USER@$(hostname)" | sudo tee -a /etc/postfix/virtual
sudo postmap /etc/postfix/virtual
sudo systemctl reload postfix
```



4. Tester :

```
echo "Test virtual" | mail -s "Test" admin@test.local
```



Bonus : Créez un catch-all pour test.local



TLS et Sécurité

🔒 Chiffrer les communications et protéger les données

Introduction au TLS

Sans TLS, vos emails circulent **en clair** sur Internet !

Tout intermédiaire (FAI, gouvernement, hacker) peut les lire.

TLS (Transport Layer Security) chiffre les communications entre serveurs SMTP.

Analogie

Sans TLS : Envoyer une carte postale (Tout le monde peut la lire en chemin)

Avec TLS : Envoyer une lettre scellée (Seul le destinataire peut l'ouvrir)

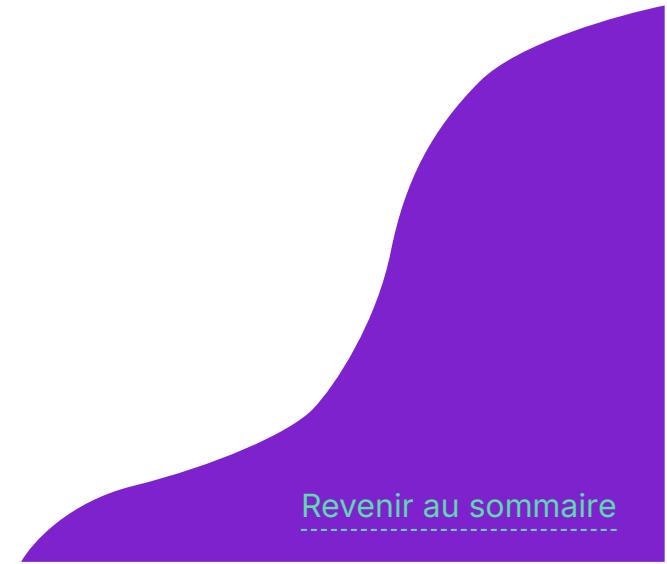


TLS vs SSL

SSL (Secure Sockets Layer) est l'ancien nom. TLS (Transport Layer Security) est la version moderne.

Versions :

- SSL 2.0 (Obsolète et vulnérable)
- SSL 3.0 (Obsolète et vulnérable)
- TLS 1.0 (Déprécié)
- TLS 1.1 (Déprécié)
- TLS 1.2 (OK mais ancien)
- **TLS 1.3 (Recommandé en 2025 ✓)**



Revenir au sommaire

Types de chiffrement SMTP

SMTP classique (port 25)

STARTTLS : Commence en clair, puis upgrade vers TLS

```
220 mail.example.com ESMTP
EHLO client.example.com
250-STARTTLS
250 HELP
STARTTLS
220 Ready to start TLS
[Connexion chiffrée]
```

SMTPS (port 465)

TLS immédiat : Chiffré dès la connexion - Obsolète depuis 1998, mais... revenu en 2018 ! - Supporté par Gmail, Outlook, etc.

Submission (port 587)

STARTTLS obligatoire : Pour les clients qui envoient des emails

Standard recommandé en 2025.

Obtenir un certificat SSL

Let's Encrypt (Gratuit !)

Let's Encrypt fournit des certificats gratuits, valables 90 jours, renouvelables automatiquement.



Installation de Certbot

```
# Ubuntu/Debian  
sudo apt install certbot
```

```
# Rocky Linux  
sudo dnf install certbot
```



🔑 Obtenir un certificat

```
sudo certbot certonly --standalone -d mail.example.com
```



Paramètres :

- certonly : Obtenir le certificat seulement (sans configurer de serveur web)
- --standalone : Serveur web temporaire (port 80 doit être libre)
- -d mail.example.com : Domaine pour lequel obtenir le certificat

Alternative si port 80 utilisé :

```
sudo certbot certonly --webroot -w /var/www/html -d mail.example.com
```



Emplacement des certificats

```
/etc/letsencrypt/live/mail.example.com/
└── cert.pem      # Certificat du serveur
    ├── chain.pem # Chaîne de certification
    ├── fullchain.pem # Certificat + chaîne (recommandé)
    └── privkey.pem # Clé privée
    README
```

Renouvellement automatique

```
# Tester le renouvellement  
sudo certbot renew --dry-run
```

```
# Configurer le cron  
sudo systemctl enable certbot-renew.timer  
sudo systemctl start certbot-renew.timer
```



Ou manuellement dans crontab :

```
0 0 * * * certbot renew --quiet --post-hook "systemctl reload postfix"
```

Certificat auto-signé (pour tests)

Si vous n'avez pas de domaine public, créez un certificat auto-signé.

 **Ne pas utiliser en production !**

Générer un certificat auto-signé

```
# Créer le répertoire  
sudo mkdir -p /etc/postfix/ssl  
  
# Générer clé et certificat  
sudo openssl req -new -x509 -days 365 -nodes \  
-out /etc/postfix/ssl/cert.pem \  
-keyout /etc/postfix/ssl/key.pem
```

Questions :

Country Name: FR
State or Province Name: Nord
Locality Name: Lille
Organization Name: MonEntreprise
Organizational Unit Name: IT
Common Name: mail.example.com
Email Address: admin@example.com



Common Name doit correspondre au hostname de votre serveur !



🔒 Permissions

```
sudo chmod 600 /etc/postfix/ssl/key.pem  
sudo chmod 644 /etc/postfix/ssl/cert.pem  
sudo chown root:root /etc/postfix/ssl/*
```



Configuration TLS dans Postfix

TLS pour les connexions entrantes (SMTPD)

```
# Certificats
smtpd_tls_cert_file = /etc/letsencrypt/live/mail.example.com/fullchain.pem
smtpd_tls_key_file = /etc/letsencrypt/live/mail.example.com/privkey.pem

# Niveau de sécurité
smtpd_tls_security_level = may
```



Niveaux de sécurité :

- none : Pas de TLS
- may : TLS si le client le supporte (recommandé pour port 25)
- encrypt : TLS obligatoire
- dane : TLS avec validation DANE
- dane-only : TLS DANE uniquement
- fingerprint : Validation par empreinte
- verify : TLS avec vérification du certificat client
- secure : TLS avec vérification stricte

🔒 Configuration recommandée pour port 25

```
smtpd_tls_security_level = may  
smtpd_tls_auth_only = yes  
smtpd_tls_loglevel = 1
```



🔒 Configuration recommandée pour port 587

```
# Port 587 = submission = TLS obligatoire  
smtpd_tls_security_level = encrypt  
smtpd_tls_auth_only = yes
```



TLS pour les connexions sortantes (SMTP)

```
# Niveau de sécurité  
smtp_tls_security_level = may  
  
# Vérifier les certificats  
smtp_tls_CAfile = /etc/ssl/certs/ca-certificates.crt  
  
# Logging  
smtp_tls_loglevel = 1
```



Versions TLS acceptées

```
# TLS 1.2 et 1.3 uniquement (recommandé 2025)
smtpd_tls_protocols = >=TLSv1.2
smtp_tls_protocols = >=TLSv1.2

# TLS 1.3 uniquement (très strict)
smtpd_tls_protocols = >=TLSv1.3
smtp_tls_protocols = >=TLSv1.3
```

Ciphers (Algorithmes de chiffrement)

```
# Ciphers forts uniquement
smtpd_tls_mandatory_ciphers = high

# Exclure les algorithmes faibles
smtpd_tls_exclude_ciphers =
    aNULL, eNULL, EXPORT, DES, RC4, MD5,
    PSK, aECDH, EDH-DSS-DES-CBC3-SHA,
    EDH-RSA-DES-CBC3-SHA, KRB5-DES-CBC3-SHA
```



Session cache

Pour améliorer les performances, Postfix peut mettre en cache les sessions TLS.

```
# Cache côté serveur  
smtpd_tls_session_cache_database = btree:${data_directory}/smtpd_scache  
smtpd_tls_session_cache_timeout = 3600s  
  
# Cache côté client  
smtp_tls_session_cache_database = btree:${data_directory}/smtp_scache  
smtp_tls_session_cache_timeout = 3600s
```



Configuration complète TLS

Dans main.cf



```
# === TLS GÉNÉRAL ===
tls_random_source = dev:/dev/urandom

# === TLS ENTRANT (SMTPD) ===
smtpd_tls_cert_file = /etc/letsencrypt/live/mail.example.com/fullchain.pem
smtpd_tls_key_file = /etc/letsencrypt/live/mail.example.com/privkey.pem
smtpd_tls_CAfile = /etc/ssl/certs/ca-certificates.crt
```

```
smtpd_tls_security_level = may
smtpd_tls_auth_only = yes
smtpd_tls_protocols = >=TLSv1.2
smtpd_tls_mandatory_protocols = >=TLSv1.2
smtpd_tls_mandatory_ciphers = high
smtpd_tls_exclude_ciphers = aNULL, eNULL, EXPORT, DES, RC4, MD5, PSK, aECDH, EDH-DSS-DES-CBC3-SHA, EDH-RSA-DES-CBC3-SHA, KRB5-DES-CBC3-SHA
```

```
smtpd_tls_session_cache_database = btree:${data_directory}/smtpd_scache
smtpd_tls_session_cache_timeout = 3600s
smtpd_tls_loglevel = 1
smtpd_tls_received_header = yes

# === TLS SORTANT (SMTP) ===
smtp_tls_security_level = may
smtp_tls_CAfile = /etc/ssl/certs/ca-certificates.crt
```

```
smtp_tls_protocols = >=TLSv1.2
smtp_tls_mandatory_protocols = >=TLSv1.2
smtp_tls_mandatory_ciphers = high
smtp_tls_session_cache_database = btree:${data_directory}/smtp_scache
smtp_tls_session_cache_timeout = 3600s
smtp_tls_loglevel = 1
```

Sécurisation du port 587 (Submission)

⚠️ Rappel : Au module 02, nous avons configuré le port 587 avec TLS **optionnel** (`smtpd_tls_security_level = may`).

Maintenant, nous allons le sécuriser en rendant TLS **obligatoire** !





Modifier master.cf

Ouvrez `/etc/postfix/master.cf` et modifiez la configuration du service `submission` :

```
submission inet n - y - - smtpd
  -o syslog_name=postfix/submission
  -o smtpd_tls_security_level=encrypt
  -o smtpd_sasl_auth_enable=yes
  -o smtpd_tls_auth_only=yes
  -o smtpd_reject_unlisted_recipient=no
  -o smtpd_client_restrictions=permit_sasl_authenticated,reject
  -o smtpd_relay_restrictions=permit_sasl_authenticated,reject
  -o milter_macro_daemon_name=ORIGINATING
```

🔍 Différences avec la configuration du module 02

`smtpd_tls_security_level=encrypt` (au lieu de `may`)

- TLS est maintenant **obligatoire** sur le port 587
- Les clients non-TLS seront rejetés

`smtpd_tls_auth_only=yes` (nouveau)

- L'authentification SASL est **impossible** sans TLS
- Force les clients à utiliser une connexion chiffrée avant de s'authentifier



Recharger Postfix

```
sudo postfix check  
sudo systemctl reload postfix
```



Tester TLS

🔍 Test avec openssl

```
openssl s_client -connect mail.example.com:25 -starttls smtp
```



Vérifiez :

```
SSL-Session:  
  Protocol : TLSv1.3  
  Cipher   : TLS_AES_256_GCM_SHA384  
  ...  
Verify return code: 0 (ok)
```



✉ Test avec telnet

```
telnet mail.example.com 25
```



```
EHLO test  
250-mail.example.com  
250-SIZE 52428800  
250-STARTTLS  
250 HELP
```

```
STARTTLS  
220 Ready to start TLS
```

 STARTTLS disponible !

🌐 Tests en ligne

SSL Labs (pour HTTPS) : <https://www.ssllabs.com/ssltest/>

CheckTLS (pour SMTP) : <https://www.checktls.com/>

MXToolbox : <https://mxtoolbox.com/diagnostic.aspx>

Forcer TLS pour certains domaines

Si vous échangez des emails sensibles avec un partenaire, forcez TLS.



Fichier /etc/postfix/tls_policy

```
# Format: domaine niveau  
  
# Forcer TLS pour ce domaine  
secure-partner.com    encrypt  
  
# Forcer TLS + vérifier certificat  
banking-partner.com   verify  
  
# Pas de TLS (si serveur ancien)  
old-system.com        none
```

Configuration

```
smtp_tls_policy_maps = hash:/etc/postfix/tls_policy
```



```
sudo postmap /etc/postfix/tls_policy  
sudo systemctl reload postfix
```



MTA-STS et DANE

🔒 MTA-STS

MTA-STS (Mail Transfer Agent Strict Transport Security) force les serveurs à utiliser TLS.

Créer /.well-known/mta-sts.txt sur mta-sts.example.com :

```
version: STSv1
mode: enforce
mx: mail.example.com
max_age: 86400
```



Ajouter un enregistrement DNS :

Type: TXT
Nom: _mta-sts.example.com
Valeur: v=STSV1; id=20250101T000000



DANE

DANE (DNS-based Authentication of Named Entities) utilise DNSSEC pour valider les certificats.

Plus complexe, nécessite DNSSEC activé sur votre domaine.



Sécurité avancée

Authentification SASL

SASL (Simple Authentication and Security Layer) permet aux clients de s'authentifier pour envoyer des emails.



🔗 Avec Dovecot (recommandé)

```
# Dans main.cf
smtpd_sasl_type = dovecot
smtpd_sasl_path = private/auth
smtpd_sasl_auth_enable = yes
smtpd_sasl_security_options = noanonymous
smtpd_sasl_local_domain = $mydomain
smtpd_sasl_tls_security_options = noanonymous
```

Socket Dovecot

Dans /etc/dovecot/conf.d/10-master.conf :

```
service auth {
    unix_listener /var/spool/postfix/private/auth {
        mode = 0660
        user = postfix
        group = postfix
    }
}
```

```
sudo systemctl restart dovecot  
sudo systemctl reload postfix
```



Restrictions d'authentification

```
smtpd_recipient_restrictions =  
    permit_mynetworks,  
    permit_sasl_authenticated,  
    reject_unauth_destination
```



Les utilisateurs authentifiés peuvent envoyer des emails même s'ils ne sont pas dans mynetworks .



Chroot et Privilèges

Par défaut, Postfix tourne en chroot pour limiter les dégâts en cas de compromission.



Dans `/etc/postfix/master.cf` :

```
# service type  private unpriv  chroot   wakeup   maxproc  command
smtp      inet  n       -       y       -       -       smtpd
```

Le `y` dans la colonne `chroot` active le chroot.

Limitations de connexions

```
# Limite de connexions simultanées par IP  
smtpd_client_connection_count_limit = 10  
  
# Taux de connexions  
smtpd_client_connection_rate_limit = 30  
anvil_rate_time_unit = 60s  
  
# Messages par connexion  
smtpd_client_message_rate_limit = 100
```



Désactiver les commandes dangereuses

```
# Désactiver VRFY (vérifier si adresse existe)  
disable_vrfy_command = yes
```

```
# Désactiver ETRN (trigger queue run)  
smtpd_etrn_restrictions = reject
```



Masquer les informations système

```
# Ne pas révéler la version  
$smtpd_banner = $myhostname ESMTP
```

```
# Headers anonymisés  
$smtpd_discard_ehlo_keywords = silent-discard, dsn
```



Firewall

🔥 UFW (Ubuntu)

```
sudo ufw allow 25/tcp  
sudo ufw allow 587/tcp  
sudo ufw allow 465/tcp
```



🔥 firewalld (Rocky Linux)

```
sudo firewall-cmd --permanent --add-service=smtp
sudo firewall-cmd --permanent --add-service=smtp-submission
sudo firewall-cmd --permanent --add-service=smtpls
sudo firewall-cmd --reload
```



Fail2ban

Bloquer les IPs qui tentent des attaques par force brute.



Installation

```
# Ubuntu/Debian  
sudo apt install fail2ban
```

```
# Rocky Linux  
sudo dnf install fail2ban
```



Configuration

Fichier `/etc/fail2ban/jail.local` :

```
[postfix-sasl]
enabled = true
port = smtp,submission,smtpls
filter = postfix-sasl
logpath = /var/log/mail.log
maxretry = 3
bantime = 3600
```

```
sudo systemctl enable fail2ban  
sudo systemctl start fail2ban
```



Surveiller les bans

```
sudo fail2ban-client status postfix-sasl
```

[Revenir au sommaire](#)



Checklist de sécurité

- TLS activé avec certificat valide
- TLS 1.2 minimum, idéalement TLS 1.3
- Ciphers forts uniquement
- Authentification SASL configurée
- Port 587 avec TLS obligatoire

- Rate limiting activé
- Fail2ban configuré
- Firewall restreint aux ports nécessaires
- Commandes dangereuses désactivées (VRFY, ETRN)
- Bannière anonymisée

- Logs activés et surveillés
- Mises à jour régulières du système
- SPF, DKIM, DMARC configurés

Exercices pratiques

🎯 Exercice 1 : Obtenir un certificat

1. Installez Certbot
2. Obtenez un certificat Let's Encrypt
3. Vérifiez qu'il est valide

🎯 Exercice 2 : Configurer TLS

1. Configurez TLS dans main.cf
2. Restreignez aux versions TLS 1.2+
3. Rechargez Postfix

🎯 Exercice 3 : Tester TLS

1. Testez avec `openssl s_client`
2. Vérifiez la version TLS utilisée
3. Testez sur <https://www.checktls.com/>

🎯 Exercice 4 : Port 587

1. Configurez le port 587 dans master.cf
2. Forcez TLS sur ce port
3. Testez avec un client mail

Points clés à retenir



Port 25 : smtpd_tls_security_level = may

Port 587 : smtpd_tls_security_level = encrypt

Versions : >= TLS 1.2 minimum

Certificats

Let's Encrypt : Gratuit, renouvelable automatiquement

Chemins : /etc/letsencrypt/live/domain/

Renouvellement : Tous les 90 jours



Sécurité

SASL : Authentification obligatoire pour envoyer

Rate limiting : Limiter les connexions abusives

Fail2ban : Bloquer les IPs attaquantes

Prochaine étape

Votre serveur est maintenant bien sécurisé ! Passons à la **surveillance et l'analyse des logs** ! 

Module suivant : Logs et surveillance →

QCM - Module 9 : TLS et sécurité

Question 1

Quelle est la différence entre SMTPS et STARTTLS ?

- A) SMTPS utilise TLS, STARTTLS utilise SSL
- B) SMTPS chiffre dès la connexion (port 465), STARTTLS upgrade une connexion (port 25/587)
- C) Il n'y a pas de différence
- D) STARTTLS est plus sécurisé



Question 2

Quel niveau TLS est recommandé pour le port 587 (submission) ?

- A) may (optionnel)
- B) encrypt (obligatoire)
- C) none (désactivé)
- D) dane (DANE uniquement)



Question 3

Où obtenir des certificats SSL gratuits et valides ?

- A) OpenSSL self-signed
- B) Let's Encrypt
- C) Acheter chez un CA commercial
- D) Copier depuis un autre serveur

Question 4

Quelle commande permet de tester manuellement STARTTLS sur le port 25 d'un serveur distant ?

- A) telnet serveur 25
- B) openssl s_client -connect serveur:25 -starttls smtp
- C) openssl s_client -connect serveur:587
- D) curl https://serveur:25

Question 5

Quel paramètre Postfix impose un niveau de chiffrement minimal pour les connexions sortantes vers d'autres serveurs ?

- A) smtpd_tls_security_level
- B) smtp_tls_security_level
- C) tlsproxy_tls_security_level
- D) smtp_tls_policy_maps

Réponses - Module 9

Question 1 : Réponse B - SMTPS (port 465) : Chiffrement dès la connexion. **STARTTLS** (port 25/587) : Connexion en clair puis upgrade TLS avec commande `STARTTLS`.

Question 2 : Réponse B - Pour le port 587 (submission), utilisez `smtpd_tls_security_level = encrypt` pour **forcer** TLS. Les clients doivent s'authentifier en chiffré !

Question 3 : Réponse B - Let's Encrypt fournit des certificats SSL **gratuits, automatiques et reconnus** par tous les navigateurs/clients mail.

Question 4 : Réponse B - `openssl s_client -connect serveur:25 -starttls smtp` négocie une session STARTTLS, affiche le certificat et les suites supportées.

Question 5 : Réponse B - `smtp_tls_security_level` s'applique au client SMTP sortant de Postfix. Avec `encrypt`, vous forcez Postfix à n'envoyer que via TLS (sinon bounce).

Exercice pratique - Module 9

🎯 Objectif

Configurer TLS sur Postfix

📋 Tâches (25 minutes)

1. Installer Certbot :

```
sudo apt install certbot # Ubuntu/Debian  
# OU  
sudo dnf install certbot # Rocky Linux
```

2. Obtenir un certificat (si domaine public) :

```
sudo certbot certonly --standalone -d mail.votredomaine.com
```



3. Configurer TLS dans Postfix :

```
sudo postconf -e "smtpd_tls_cert_file = /etc/letsencrypt/live/mail.votredomaine.com/fullchain.pem"
sudo postconf -e "smtpd_tls_key_file = /etc/letsencrypt/live/mail.votredomaine.com/privkey.pem"
sudo postconf -e "smtpd_tls_security_level = may"
sudo postconf -e "smtpd_tls_protocols = !SSLv2, !SSLv3, !TLSv1, !TLSv1.1"
sudo postconf -e "smtp_tls_security_level = may"
sudo postconf -e "smtp_tls_protocols = !SSLv2, !SSLv3, !TLSv1, !TLSv1.1"
```

4. Configurer le port 587 :

```
# Dans /etc/postfix/master.cf, décommenter/ajouter :  
submission inet n - y - - smtpd  
-o smtpd_tls_security_level=encrypt  
-o smtpd_sasl_auth_enable=yes
```

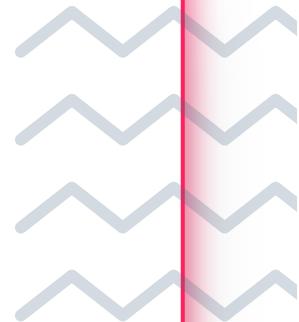
5. Recharger et tester :

```
sudo systemctl reload postfix  
openssl s_client -connect localhost:587 -starttls smtp
```



Bonus : Configurez le renouvellement automatique avec certbot





DKIM, SPF et DMARC

🔒 Authentifier vos emails et améliorer votre délivrabilité

Introduction

Sans SPF, DKIM et DMARC, vos emails risquent de :

- Être marqués comme spam
- Être rejetés par **Gmail, Outlook, etc.**
- Permettre l'usurpation de votre domaine (spoofing)

En 2025, ces trois standards sont **obligatoires** pour toute infrastructure mail professionnelle.



Les 3 piliers de l'authentification

SPF : "Quels serveurs peuvent envoyer des emails pour mon domaine ?"

DKIM : "Cet email vient vraiment de moi (signature cryptographique)"

DMARC : "Comment traiter les emails qui échouent SPF/DKIM ?"

Analogie

SPF = Liste des facteurs autorisés

- **DKIM** = Cachet officiel de la poste
- **DMARC** = Politique de traitement des lettres suspectes

SPF (Sender Policy Framework)

Qu'est-ce que SPF ?

SPF est un enregistrement DNS qui liste les serveurs autorisés à envoyer des emails pour votre domaine.

🔍 Comment ça marche ?

1. Serveur spammer.com envoie un email prétendant venir de example.com
2. Serveur destinataire vérifie l'enregistrement SPF de example.com
3. L'IP spammer.com n'est pas dans la liste → Rejet !

Format d'un enregistrement SPF

```
v=spf1 <mécanismes> <qualifiers> <all>
```

Exemple simple

```
v=spf1 mx a ip4:203.0.113.10 -all
```

The screenshot shows the OVH DNS management interface. The left sidebar lists services for the domain 'jimmylan.fr'. The main panel displays the 'Zone DNS' configuration for this domain. A table lists various DNS records, including:

Domaine	TTL	Type	Cible
jimmylan.fr.	0	NS	ns1.ovh.net.
mail.jimmylan.fr.	0	CNAME	ssl0.ovh.net.
smtplib.jimmylan.fr.	0	CNAME	ssl0.ovh.net.
pop3.jimmylan.fr.	0	CNAME	ssl0.ovh.net.
imap.jimmylan.fr.	0	CNAME	ssl0.ovh.net.
jimmylan.fr.	600	SPF	v=spf1 mx@testmail.jimmylan.fr include:mx.ovh.com ~all
atlas.jimmylan.fr.	0	A	51.68.224.131
jimmylan.fr.	0	A	76.76.21.21
www.jimmylan.fr.	0	CNAME	cname.vercel-dns.com.
testmail.jimmylan.fr.	0	A	51.68.224.131

On the right, there are buttons for adding, modifying, and deleting records, as well as viewing history and resetting the zone. A 'Guides' section is also present.

Je vous ai mis une capture d'écran de l'enregistrement SPF de mon domaine 'jimmylan.fr'.

Décryptage : v=spf1 (Version SPF)

- mx (Les serveurs MX du domaine peuvent envoyer)
- a (Le serveur A du domaine peut envoyer) : testmail.jimmylan.fr
- ip4:203.0.113.10 (Cette IP peut envoyer) - ~all (Rejeter tout le reste)

J'ai quand même mis le ~all pour que vous voyez que c'est possible.



🎯 Mécanismes SPF

a : Autorise l'enregistrement A du domaine

```
v=spf1 a -all
```

mx : Autorise les serveurs MX du domaine

```
v=spf1 mx -all
```

ip4:IP : Autorise une IP ou plage IPv4

```
v=spf1 ip4:203.0.113.10 ip4:203.0.113.0/24 -all
```

ip6:IP : Autorise une IP IPv6

```
v=spf1 ip6:2001:db8::1 -all
```

include:domaine : Inclut le SPF d'un autre domaine

```
v=spf1 include:_spf.google.com -all
```

(Utile si vous utilisez Gmail pour envoyer)

exists:domaine : Vérifie l'existence d'un enregistrement A (Rarement utilisé)

Qualifiers

+ : PASS (autorisé) - défaut

```
v=spf1 +mx -all  
# Équivalent à  
v=spf1 mx -all
```

- : FAIL (rejeté)

```
v=spf1 mx -all
```

~ : SOFTFAIL (suspect mais pas rejeté)

```
v=spf1 mx ~all
```

? : NEUTRAL (neutre, pas de recommandation)

```
v=spf1 mx ?all
```

💡 Quelle fin choisir ?

-all : Strict (recommandé si vous contrôlez tous vos serveurs)

- **~all** : Permissif (si vous avez peur de bloquer des emails légitimes)
- **?all** : Très permissif (déconseillé)

En 2025, utilisez **-all** si possible !

Créer votre enregistrement SPF

Étape 1 : Lister vos serveurs d'envoi

- Votre serveur Postfix : mail.example.com (IP: 203.0.113.10)
- Gmail (G Suite) pour certains utilisateurs
- Service marketing (ex: Mailchimp)

Étape 2 : Construire l'enregistrement

```
v=spf1  
a:mail.example.com  
ip4:203.0.113.10  
include:_spf.google.com  
include:servers.mcsv.net  
-all
```

🌐 Étape 3 : Ajouter l'enregistrement DNS

Type : TXT/SPF - Nom : example.com (ou @) - Valeur :

```
v=spf1 a mx ip4:203.0.113.10 -all
```

⌚ Étape 4 : Attendre la propagation DNS

Cela peut prendre de quelques minutes à 48 heures.

✓ Étape 5 : Tester

```
# Vérifier l'enregistrement SPF  
dig example.com TXT +short | grep spf  
  
# Ou avec nslookup  
nslookup -type=TXT example.com
```

Outils en ligne : <https://mxtoolbox.com/spf.aspx> - <https://www.kitterman.com/spf/validate.html>

Renouvellement automatique prévu en janv. 2026

Informations générales Zone DNS Serveurs DNS Redirection DynHost GLUE DS Records

Ajouter une entrée Modifier en mode texte Modifier le TTL par défaut Voir l'historique de ma zone DNS Réinitialiser ma zone DNS

Domaine	TTL	Type	Cible
jimmylan.fr.	0	MX	1 testmail.jimmylan.fr.
_dmarc.jimmylan.fr.	0	DMARC	v=DMARC1;p=quarantine;sp=none;aspf=r; v=DKIM1;h=sha256;p=MIBjANBkgkhkG9w0BAQFAOCAQ8 AMIIICgKCQAQEAIV03/E0hpuXaoC/EuJG5R7NpbkYTALZ1f wZ2X49YTFXfmrxXF99HYHCzj40g9j9rsLU1F56oLw2eV4 SnugUG4eQMaZVW0D2wnt7aQoBwLUBZLgiH3mJD7USTOTUa CxQR3BL6y8nMNg8msvDHn2Kycjze6q9NxEgbzAKsdg+I cqDF/pj23mrNZ12zmfjzaObq/FKWKbbqvas8ddsl9jScifFTS EuLz+yUdcqCqB8Rfcu+35nZEMVfEp7BUhGN2zpJQ2tH mvu41tYO16hA3ajSUT7fbql3qtwuJ8XnMLRHfuHKhWcWE3td 5EwigcGf4QIDAQAB
mail._domainkey.jimmylan.fr.	0	DKIM	

OK

Un petit gif pour illustrer l'enregistrement SPF



Limites de SPF

⚠ Maximum 10 lookups DNS dans un enregistrement SPF !

Chaque include: compte comme un lookup.

Solution : Remplacer les `include:` par des IPs directes quand possible.



DKIM (DomainKeys Identified Mail)

Qu'est-ce que DKIM ?

DKIM signe cryptographiquement vos emails pour prouver qu'ils viennent vraiment de vous.



🔍 Comment ça marche ?

1. Postfix signe l'email avec une clé privée
2. La signature est ajoutée dans les headers (DKIM-Signature:)
3. Destinataire récupère la clé publique via DNS
4. Vérifie la signature

Analogie : C'est comme un cachet de cire sur une lettre médiévale. Si le cachet est brisé, la lettre a été modifiée.

Installation d'OpenDKIM

```
# Ubuntu/Debian  
sudo apt install opendkim opendkim-tools
```

```
# Rocky Linux  
sudo dnf install opendkim
```



Configuration d'OpenDKIM

Fichier /etc/opendkim.conf

```
# Mode  
Mode sv  
  
# Socket pour Postfix  
Socket inet:8891@localhost  
  
KeyTable /etc/opendkim/KeyTable # Table des clés  
SigningTable /etc/opendkim/SigningTable # Table des signatures  
ExternalIgnoreList /etc/opendkim/TrustedHosts # Liste des serveurs autorisés  
InternalHosts /etc/opendkim/TrustedHosts # Liste des serveurs autorisés
```

Le fichier en entier :

```
# Mode
Mode sv # Mode de fonctionnement

# Socket pour Postfix
Socket inet:8891@localhost

# UMask
UMask 007 # Permet de limiter les permissions des fichiers
007

# Logging
SysLog yes # Log dans le fichier de log de Postfix
SyslogSuccess yes # Log les succès
LogWhy yes # Log les raisons des rejets

KeyTable /etc/opendkim/KeyTable # Table des clés
SigningTable /etc/opendkim/SigningTable # Table des signatures
ExternalIgnoreList /etc/opendkim/TrustedHosts # Liste des serveurs autorisés
InternalHosts /etc/opendkim/TrustedHosts # Liste des serveurs autorisés

# Canonicalisation
Canonicalization relaxed/simple # Canonicalisation des emails

PidFile /run/opendkim/opendkim.pid # PID file

# Autres
AutoRestart yes # Redémarrer le service si il crash
AutoRestartRate 10/1h # Redémarrer le service si il crash 10 fois en 1 heure
```



🔑 Générer les clés DKIM

```
# Créer le répertoire  
sudo mkdir -p /etc/opendkim/keys/jimmylan.fr  
  
# Générer la paire de clés  
sudo opendkim-genkey -b 2048 -d jimmylan.fr -D /etc/opendkim/keys/jimmylan.fr -s mail -v
```



Paramètres :

- -b 2048 : Taille de la clé (2048 bits recommandé en 2025)
- -d jimmylan.fr : Domaine
- -D /path : Répertoire de sortie
- -s mail : Sélecteur
- -v : Verbose

Cela crée deux fichiers :

- mail.private : Clé privée (garde secret !)
- mail.txt : Clé publique (à publier en DNS)



Permissions

Vous pouvez aussi utiliser la commande suivante pour changer les permissions des fichiers :

```
sudo chown opendkim:opendkim /etc/opendkim/keys/example.com/mail.private  
sudo chmod 600 /etc/opendkim/keys/example.com/mail.private
```

Maintenant nous allons créer les fichiers de configuration pour OpenDKIM.

Nous avons :

```
- /etc/opendkim/KeyTable  
- /etc/opendkim/SigningTable  
- /etc/opendkim/TrustedHosts
```

Pourquoi ces fichiers ?

- KeyTable : Table des clés = correspondance entre le domaine et la clé
- SigningTable : Table des signatures = correspondance entre le domaine et la signature
- TrustedHosts : Liste des serveurs autorisés = liste des serveurs autorisés à envoyer des emails via DKIM (IP, domaine, etc.)



Cas spécifique :

On a un utilisateur qui s'appelle john doe

Donc on veut forcement johndoe@jimmylan.fr

Mais on a fais un enregistrement MX et A sur mail.jimmylan.fr

Donc il faut quand même préciser que on veux generer des clef pour le nom de domaine en entier et pas uniquement le sous domaine.

Fichier /etc/opendkim/KeyTable

```
mail._domainkey.jimmylan.fr jimmylan.fr:mail:/etc/opendkim/keys/jimmylan.fr/mail.private # la clé mail._domainkey.jimmylan.fr est stockée dans le  
fichier /etc/opendkim/keys/jimmylan.fr/mail.private
```

Format: selector._domainkey.domain domain:selector:keyfile

Fichier /etc/opendkim/SigningTable

```
*@jimmylan.fr mail._domainkey.jimmylan.fr # tous les emails de jimmylan.fr seront signés par la clé mail._domainkey.jimmylan.fr  
john Doe@jimmylan.fr mail._domainkey.jimmylan.fr # tous les emails de johnDoe@jimmylan.fr seront signés par la clé mail._domainkey.jimmylan.fr
```

Format: pattern key

Fichier /etc/opendkim/TrustedHosts

```
127.0.0.1 # localhost  
localhost # le mapping de localhost vers 127.0.0.1  
testmail.jimmylan.fr # votre serveur mail  
51.68.224.131 # votre IP publique
```

Démarrer OpenDKIM

```
sudo systemctl enable opendkim  
sudo systemctl start opendkim  
sudo systemctl status opendkim
```



Intégration avec Postfix

⚙️ Configuration dans main.cf

```
# DKIM signing via OpenDKIM
smtpd_milters = inet:localhost:8891
non_smtpd_milters = inet:localhost:8891
milter_default_action = accept
milter_protocol = 6
```

On pourrait effectivement utiliser un socket UNIX ou un socket local pour plus de sécurité.

Dans le cadre de cette formation, on va rester sur le socket inet.

```
smtpd_milters = unix:/var/run/opendkim/opendkim.sock
non_smtpd_milters = unix:/var/run/opendkim/opendkim.sock
```

```
sudo systemctl reload postfix
```



Publier la clé publique DKIM en DNS

Récupérer la clé publique

```
sudo cat /etc/opendkim/keys/example.com/mail.txt
```



Contenu :

```
mail._domainkey IN TXT ( "v=DKIM1; h=sha256; k=rsa; "
"p=MIIBIjANBgkqhkiG9w0BAQEFAOCAQ8AMIIBCgKCAQEA...""
"..." ) ; ----- DKIM key mail for example.com
```

Ajouter l'enregistrement DNS

Type : TXT/DKIM

Nom : mail._domainkey.jimmylan.fr

Valeur :

```
v=DKIM1; h=sha256; p=MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA...
```

⚠ **Important :** Supprimez les guillemets et concaténez les lignes !

Il faut que la clef soit en une seule ligne et sans guillemets.

✓ Vérifier l'enregistrement DNS

```
dig mail._domainkey.jimmylan.fr TXT +short
```



The screenshot shows the OVH DNS management interface for the domain `jimmylan.fr`. The left sidebar lists various services, with `jimmylan.fr` selected. The main panel displays the DNS zone configuration for `jimmylan.fr`. The `Zone DNS` tab is active, showing three entries:

Domaine	TTL	Type	Cible
<code>jimmylan.fr.</code>	0	MX	1 testmail.jimmylan.fr.
<code>_dmarc.jimmylan.fr.</code>	0	DMARC	v=DMARC1;p=quarantine;sp=none;aspf=r; v=DKIM1;h=sha256;p=MIBjANBkgkhkG9w0BAQFAOCAQ8 AMIIICgKCQAQEAIV03/E0hpuXaoC/EuJG5R7NpbkYTAlZ1f wZ2Xq9YTFXfmrxXF99HYCzj40g9j9rsLU1F56oLw2eV4 SnugUG4eQMgZVW0D02wnt7QoBwUUBZLgiH3mJD7USTOTuA CxQR3BL6y8nMNg8msvDHvNzKycZ6eq9NxEgbzAKsdg+I qDF/pj23mrNZ12zmfjgjA0bjp/FKWKbbqvas8ddsl9jScIdFTS EuLz+yUdcQgB8Rfcu35nZEMVfEp7BuhGN2zpJQ2tH mvu41tYO16hA3ajSUT7fbql3qtwuJ8XnMLRHfuHKhWcWE3td 5EwigcGf4QIDAQAB
<code>mail._domainkey.jimmylan.fr.</code>	0	DKIM	

On the right, there are buttons for managing entries: `Ajouter une entrée`, `Modifier en mode texte`, `Modifier le TTL par défaut`, `Voir l'historique de ma zone DNS`, and `Réinitialiser ma zone DNS`. A sidebar titled `Guides` provides links to `Zone DNS`.

Un petit gif pour illustrer la vérification de l'enregistrement DNS



Outils en ligne :

- <https://mxtoolbox.com/dkim.aspx>



Tester DKIM

Envoyer un email de test

```
echo "Test DKIM" | mail -s "Test DKIM" -r "johndoe@jimmylan.fr" check-auth@verifier.port25.com
```

A auth-results@verifier.port25.com

Boîte de...romed.cloud avant-hier à 17:50

Authentication Report

À : johndoe@andromed.cloud

This message is an automatic response from Port25's authentication verifier service at Verifier.port25.com. The service allows email senders to perform a simple check of various sender authentication mechanisms. It is provided free of charge, in the hope that it is useful to the email community. While it is not officially supported, we welcome any feedback you may have at <Verifier-feedback@port25.com>.

Thank you for using the verifier,

The Port25 Solutions, Inc. team

Summary of Results

SPF check: pass

"iprev" check: pass

DKIM check: pass

tout est ok ! 🎉

Details:

HELO hostname: mail.andromed.cloud

Source IP: 51.68.120.60

mail-from: johndoe@andromed.cloud

SPF check details:

Result: pass

ID(s) verified: smtp.mailfrom=johndoe@andromed.cloud

DNS record(s):

andromed.cloud. 164 IN TXT "1^{www.andromed.cloud}"

andromed.cloud. 164 IN TXT "v=spf1 mx a:mail.andromed.cloud include:mx.ovh.com ~all"

andromed.cloud. 164 IN MX 10 mail.andromed.cloud.

mail.andromed.cloud. 165 IN A 51.68.120.60

Check details:

pass (matches vps-02de336b.vps.ovh.net)

ID(s) verified: policy.iprev=51.68.120.60



Vous pouvez aussi utiliser mail-tester.com pour tester DKIM.

🔍 **Vous devriez avoir ces résultats :**

Je vous fournis des images ci dessous

Ouah ! Parfait, vous pouvez envoyer

Votre note :

10/10



La note de 10/10



✓ Cliquez ici pour afficher votre message	✓
✗ SpamAssassin vous aime	✓
✗ Vous n'êtes pas parfaitement authentifié	✗
✗ Votre message pourrait être amélioré	✗
✗ Votre serveur n'est pas blocklisté	✓

Votre joli total : 10/10



Le recap de la note de 10/10

^ Vous n'êtes pas parfaitement authentifié



Nous vérifions que votre serveur d'envoi est authentifié

✓ [SPF] Votre serveur **51.68.120.60** est authentifié pour utiliser **johndoe@andromed.cloud**



✓ Votre signature DKIM est valide



✓ Votre message a réussi le test DMARC



✓ Votre reverse DNS ne correspond pas avec votre domaine d'envoi.



✓ Votre nom de domaine **andromed.cloud** est rattaché à un serveur mail.



✓ Votre nom d'hôte **mail.andromed.cloud** est rattaché à un serveur.



Le test de DKIM avec mail-tester.com



^ SpamAssassin vous aime



Le fameux filtre anti-spam **SpamAssassin**. La note : 0.7.

Une note en dessous de -5 est considérée comme du spam.

-0.1	DKIM_SIGNED	Message has a DKIM or DK signature, not necessarily valid Cette règle est automatiquement appliquée si votre email contient une signature DKIM mais d'autres règles seront également ajoutées si la signature DKIM est valide. Voir immédiatement après.
0.1	DKIM_VALID	Message has at least one valid DKIM or DK signature Génial ! Votre signature est valide
0.1	DKIM_VALID_AU	Message has a valid DKIM or DK signature from author's domain Génial ! Votre signature est valide et provient de votre nom de domaine
0.1	DKIM_VALID_EF	Message has a valid DKIM or DK signature from envelope-from domain
-1.499	FROM_FMBLA_NEWDOM	From domain was registered in last 7 days
2	RCVD_IN_RP_SAFE	Sender is in Return Path Safe (trusted relay)
-0.001	SPF_HELO_NONE	SPF: HELO does not publish an SPF Record
0.001	SPF_PASS	SPF: sender matches SPF record Super! Votre enregistrement SPF est valide



Vérifier la signature

Dans votre boîte mail, ouvrez l'email → "Afficher l'original" → Cherchez :

```
DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/simple;  
d=example.com; s=mail; t=1234567890;  
h=from:to:subject:date:message-id;  
bh=...;  
b=...
```

Et vérifiez :

```
dkim=pass header.d=jimmylan.fr
```

 DKIM fonctionne !

Rotation des clés DKIM

Bonne pratique : Changer les clés DKIM tous les 6-12 mois.



Processus :

1. Générer une nouvelle paire de clés avec un nouveau sélecteur
2. Publier la nouvelle clé publique en DNS
3. Configurer OpenDKIM pour utiliser la nouvelle clé
4. Attendre 48h (propagation DNS)
5. Supprimer l'ancienne clé du DNS



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

Qu'est-ce que DMARC ?

DMARC combine SPF et DKIM et définit une **politique** : que faire si SPF ou DKIM échoue ?

🔍 Comment ça marche ?

1. Email arrive chez le destinataire
2. Vérification SPF : ou
3. Vérification DKIM : ou
4. Consulte la politique DMARC
5. Applique l'action (accepter, quarantaine, rejeter)
6. Envoie un rapport au propriétaire du domaine

Format d'un enregistrement DMARC

```
v=DMARC1; p=reject; rua=mailto:dmarc@jimmylan.fr; ruf=mailto:dmarc@jimmylan.fr; pct=100
```





Tags DMARC

v : Version (toujours DMARC1)

p : Politique (none, quarantine, reject)

rua : Adresse pour les rapports agrégés

ruf : Adresse pour les rapports forensiques

pct : Pourcentage d'emails concernés (0-100)

adkim : Alignement DKIM (s=strict, r=relaxed)

aspf : Alignement SPF (s=strict, r=relaxed)

sp : Politique pour les sous-domaines

🎯 Politiques DMARC

none : Monitoring uniquement (aucune action)

```
v=DMARC1; p=none; rua=mailto:dmarc@example.com
```

Utilisez ça au début pour surveiller !



quarantine : Marquer comme spam

```
v=DMARC1; p=quarantine; ruamailto:dmarc@example.com
```



reject : Rejeter complètement

```
v=DMARC1; p=reject; ruamailto:dmarc@example.com
```

Le plus strict ! À utiliser quand vous êtes sûr de votre config.



Créer votre enregistrement DMARC

Étape 1 : Démarrer avec "none"

```
v=DMARC1; p=none; rua=mailto:dmarc-reports@example.com; pct=100
```



🌐 Étape 2 : Ajouter l'enregistrement DNS

Type : TXT

Nom : _dmarc.example.com

Valeur :

```
v=DMARC1; p=none; rua=mailto:dmarc-reports@example.com; pct=100
```

ovh.com/manager/#/web/domain/jimmylan.fr/zone

Tableau de bord Bare Metal Cloud Hosted Private Cloud Public Cloud Web Cloud Télécom Sunrise Marketplace

Version classique Version beta Français Jimmylan Surquin

jimmylan.fr

Renouvellement automatique prévu en janv. 2026

Roadmap & Changelog Actions

Informations générales Zone DNS Serveurs DNS Redirection DynHost GLUE DS Records

Ajouter une entrée Modifier en mode texte Modifier le TTL par défaut Voir l'historique de ma zone DNS Réinitialiser ma zone DNS

Tous Recherche domaine...

Domaine	TTL	Type	Cible	Actions
jimmylan.fr.	0	MX	1 testmail.jimmylan.fr.	...
_dmarc.jimmylan.fr.	0	DMARC	v=DMARC1;p=quarantine;sp=none;aspf=r;	...
mail._domainkey.jimmylan.fr.	0	DKIM	CxQR3BL6y8nMNg8msvDHV6nzKYcpZee6qx9NzEgebzAK4sdg+I qDF/pj2smrN1ZLzzmfgzjAOhq/FfKWBbgvasBddsu9zScidFTSG Eulz+yUdtqCqB8aRfcu+35nZEfMVEp2TbUhGN2zpJQ2ttHp mvu41tYO16HpaA3aJSU7HbqL3qtwuJ8XnMLRHfuHKWcWE3td 5EwrjcGfQIDAQAB;	...

Guides Zone DNS

1 2 3 4 5

10 Page 3 / 3 OK

Revenir au sommaire

Étape 3 : Surveiller les rapports

Les serveurs qui reçoivent vos emails envoient des rapports XML à `dmarc-reports@example.com` .

Format des rapports : XML (pas très lisible...)

Solution : Utilisez un service comme :

- <https://dmarc.postmarkapp.com/> (gratuit)
- <https://dmarcian.com/>
- <https://mxtoolbox.com/dmarc.aspx>

🔒 Étape 4 : Passer à "quarantine" puis "reject"

Après 1-2 semaines de monitoring sans problème :

```
v=DMARC1; p=quarantine; rua=mailto:dmarc-reports@example.com; pct=100
```

Après 1 mois :

```
v=DMARC1; p=reject; ruamailto:dmarc-reports@example.com; pct=100
```



Tester DMARC

```
dig _dmarc.example.com TXT +short
```

[Revenir au sommaire](#)



Outils en ligne :

- <https://mxtoolbox.com/dmarc.aspx>
- <https://dmarcian.com/dmarc-inspector/>

Configuration complète SPF + DKIM + DMARC

🌐 Enregistrements DNS

Enregistrement SPF :

```
Type: TXT  
Nom: example.com (ou @)  
Valeur: v=spf1 a mx ip4:203.0.113.10 -all
```



Enregistrement DKIM :

```
Type: TXT
Nom: mail._domainkey.example.com
Valeur: v=DKIM1; h=sha256; k=rsa; p=MIIIBIjAN...
```



Enregistrement DMARC :

```
Type: TXT
Nom: _dmarc.example.com
Valeur: v=DMARC1; p=reject; rua=mailto:dmarc@example.com; pct=100
```



Vérification complète

 **Test via email**

Envoyez un email à : check-auth@verifier.port25.com

Vous recevrez un rapport complet sur SPF, DKIM, DMARC !



✓ Test via outils

- <https://www.mail-tester.com/> (note sur 10) > Vous devez avoir un score de 10/10
- <https://mxtoolbox.com/emailhealth/>

Troubleshooting

✗ Problème : SPF échoue

Cause 1 : Enregistrement SPF incorrect

```
dig jimmylan.fr TXT +short | grep spf
```

Vérifiez la syntaxe !



Cause 2 : Email envoyé depuis une IP non autorisée

Solution : Ajoutez l'IP au SPF ou utilisez un relais autorisé



✖ Problème : DKIM échoue

Cause 1 : Clé publique pas en DNS ou incorrecte

```
dig mail._domainkey.jimmylan.fr TXT +short
```



Cause 2 : OpenDKIM ne signe pas

```
sudo systemctl status opendkim
sudo tail -f /var/log/mail.log | grep dkim
# ou
sudo tail -f /var/log/opendkim/opendkim.log | grep dkim
# ou
sudo journalctl -u opendkim -f
```



Cause 3 : Permissions sur la clé privée

```
sudo chmod 600 /etc/opendkim/keys/jimmylan.fr/mail.private
sudo chown opendkim:opendkim /etc/opendkim/keys/jimmylan.fr/mail.private
```

✖ Problème : DMARC échoue

Cause : SPF ET DKIM échouent tous les deux

Solution : Fixez au moins un des deux !

Cas d'usage avancés

Plusieurs domaines

OpenDKIM :

```
# KeyTable  
mail._domainkey.example.com example.com:mail:/etc/opendkim/keys/example.com/mail.private  
mail._domainkey.example.org example.org:mail:/etc/opendkim/keys/example.org/mail.private  
  
# SigningTable  
*@example.com mail._domainkey.example.com  
*@example.org mail._domainkey.example.org
```

Sous-domaines

DMARC avec politique pour sous-domaines :

```
v=DMARC1; p=reject; sp=quarantine; rua=mailto:dmarc@example.com
```

- p=reject : Politique pour example.com
- sp=quarantine : Politique pour *.example.com

Points clés à retenir



SPF

- Enregistrement TXT sur example.com
- Liste les serveurs autorisés
- Finir par -all (strict) ou ~all (permissif)
- Maximum 10 lookups DNS

DKIM

- Signature cryptographique des emails
- Clé privée sur le serveur (OpenDKIM)
- Clé publique en DNS (selector._domainkey.domain)
- Rotation des clés tous les 6-12 mois

DMARC

- Combine SPF + DKIM
- Définit la politique en cas d'échec
- Enregistrement TXT sur _dmarc.example.com
- Commencer par p=none , finir par p=reject

Progression recommandée

1. **Semaine 1** : Configurer SPF et DKIM
2. **Semaine 2** : Activer DMARC avec p=none
3. **Semaine 4** : Passer à p=quarantine
4. **Semaine 8** : Passer à p=reject

Exercices pratiques

🎯 Exercice 1 : SPF

1. Créez un enregistrement SPF pour votre domaine
2. Publiez-le en DNS
3. Testez avec `dig` et un outil en ligne

🎯 Exercice 2 : DKIM

1. Installez OpenDKIM
2. Générez une paire de clés
3. Configurez Postfix pour signer
4. Publiez la clé publique en DNS
5. Envoyez un email de test et vérifiez la signature

🎯 Exercice 3 : DMARC

1. Créez un enregistrement DMARC avec p=none
2. Publiez-le en DNS
3. Envoyez des emails et consultez les rapports

🎯 Exercice 4 : Test complet

1. Envoyez un email à check-auth@verifier.port25.com
2. Analysez le rapport reçu
3. Testez sur <https://www.mail-tester.com/>
4. Visez un score de 10/10 !

Prochaine étape

SPF, DKIM et DMARC sont configurés ! Passons maintenant au **chiffrement TLS** pour sécuriser les communications ! 

Module suivant : TLS et sécurité →



QCM - Module 8 : DKIM, SPF et DMARC

Question 1

À quoi sert SPF ?

- A) Chiffrer les emails
- B) Authentifier le serveur expéditeur
- C) Filtrer le spam
- D) Compresser les pièces jointes

Question 2

Où se trouve la signature DKIM dans un email ?

- A) Dans le corps du message
- B) Dans les en-têtes (DKIM-Signature)
- C) Dans le fichier de configuration
- D) Dans les logs



Question 3

Que signifie p=reject dans un enregistrement DMARC ?

- A) Rejeter tous les emails
- B) Rejeter les emails qui échouent SPF/DKIM
- C) Mettre en quarantaine
- D) Accepter mais signaler

Question 4

Quel tag d'un enregistrement DMARC indique l'adresse de réception des rapports agrégés ?

- A) p
- B) rua
- C) adkim
- D) sp

Question 5

Dans un enregistrement SPF, que signifie le mécanisme ~all placé en fin de règle ?

- A) Refuser toute adresse non listée (fail dur)
- B) Autoriser tout le monde (pass)
- C) Acceptation conditionnelle avec alerte (softfail)
- D) Ignorer la règle SPF

Réponses - Module 8

Question 1 : Réponse B - SPF (Sender Policy Framework) permet de vérifier que l'IP qui envoie un email est **autorisée** à le faire pour ce domaine.

Question 2 : Réponse B - La signature DKIM est dans les **en-têtes** de l'email (champ DKIM-Signature:). Elle signe cryptographiquement le message.

Question 3 : Réponse B - p=reject demande aux serveurs destinataires de **REJETER** les emails qui échouent les vérifications SPF ou DKIM. C'est la politique la plus stricte.

Question 4 : Réponse B - rua=mailto:.... définit la boîte qui recevra les rapports quotidiens agrégés. Vous pouvez ajouter plusieurs adresses séparées par des virgules.

Question 5 : Réponse C - ~all applique un **softfail** : les serveurs destinataires sont encouragés à marquer l'email comme suspect mais pas à le rejeter systématiquement (~all serait un fail dur).



Exercice pratique - Module 8

🎯 Objectif

Configurer SPF, DKIM et DMARC

📋 Tâches (30 minutes)

1. Créer un enregistrement SPF :

```
; Dans votre zone DNS  
votredomaine.local. IN TXT "v=spf1 mx a ip4:VOTRE_IP ~all"
```



2. Installer OpenDKIM :

```
sudo apt install opendkim opendkim-tools # Ubuntu/Debian  
# OU  
sudo dnf install opendkim # Rocky Linux
```

3. Générer les clés DKIM :

```
sudo mkdir -p /etc/opendkim/keys/votredomaine.local  
sudo opendkim-genkey -D /etc/opendkim/keys/votredomaine.local/ -d votredomaine.local -s default  
sudo chown -R opendkim:opendkim /etc/opendkim/keys/
```

4. Afficher la clé publique :

```
sudo cat /etc/opendkim/keys/votredomaine.local/default.txt
```

5. Ajouter l'enregistrement DNS :

```
default._domainkey.votredomaine.local. IN TXT "v=DKIM1; k=rsa; p=VOTRE_CLE_PUBLIQUE"
```

6. Créer l'enregistrement DMARC :

```
_dmarc.votredomaine.local. IN TXT "v=DMARC1; p=none; rua=mailto:dmarc@votredomaine.local"
```

7. Tester : Envoyez un email et vérifiez les en-têtes DKIM

Bonus : Testez avec <https://www.mail-tester.com/>



Dovecot - Serveur IMAP/POP3

Intégration avec Postfix

Configuration complète d'un serveur de messagerie

Qu'est-ce que Dovecot ?

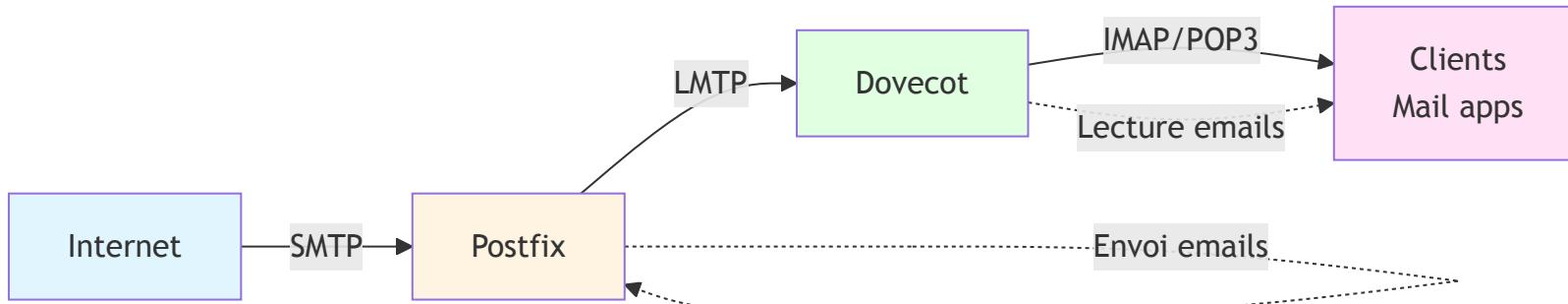
Dovecot est un serveur IMAP et POP3 open-source

Rôle dans l'écosystème email

- **Postfix** : MTA (Mail Transfer Agent) - Envoie et reçoit les emails
- **Dovecot** : MDA (Mail Delivery Agent) - Stocke et permet de récupérer les emails

Complémentarité Postfix + Dovecot

Workflow complet



✨ Pourquoi Dovecot ?

Avantages

- ⚡ **Performances** : Très rapide, même avec des milliers de boîtes
- 🔒 **Sécurité** : Support complet de SSL/TLS, authentification robuste
- 📊 **Scalabilité** : Gère de petites à très grandes installations
- 🔧 **Flexibilité** : Nombreuses options de stockage et d'authentification
- 📱 **Moderne** : Support IMAP IDLE, push notifications

Installation sur Debian/Ubuntu

Installation des paquets

```
# Mise à jour du système  
apt update && apt upgrade -y  
  
# Installation de Dovecot avec IMAP et POP3  
apt install -y dovecot-core dovecot-imapd dovecot-pop3d  
  
# Installation des modules complémentaires  
apt install -y dovecot-lmtpd dovecot-mysql dovecot-sieve
```



Petite aparté : LMTPD et Dovecot-Sieve

LMTPD

Définition simple : Reçoit les emails de Postfix et les dépose dans les boîtes aux lettres.

Analogie :

Postfix = facteur qui apporte les sacs de courrier

LMTPD = personne dans le centre de tri qui ouvre les sacs et range chaque lettre dans la bonne case

Dovecot-Sieve

Définition simple : Langage de règles qui trie automatiquement les emails (spam, newsletter, etc.)

Analogie :

Sieve = assistant personnel qui lit chaque enveloppe et place chaque email dans le bon dossier

Sans Sieve, tout tombe en vrac dans la même boîte

📦 Installation sur CentOS/RHEL

Installation avec dnf/yum

```
# Installation sur CentOS/RHEL 8+
dnf install -y dovecot dovecot-mysql dovecot-pigeonhole
```



```
# Démarrage et activation  
systemctl start dovecot  
systemctl enable dovecot
```

```
# Vérification de l'installation  
dovecot --version
```

Architecture de Dovecot

Composants principaux

Services

- dovecot : Processus maître
- imap : Serveur IMAP
- pop3 : Serveur POP3
- lmtp : Local Mail Transfer Protocol
- auth : Service d'authentification
- lda : Local Delivery Agent

Structure des fichiers

Fichiers de configuration

```
/etc/dovecot/
└── dovecot.conf          # Configuration principale
    ├── conf.d/           # Configurations modulaires
    └── 10-auth.conf       # Authentification
    ├── 10-mail.conf       # Stockage des emails
    ├── 10-master.conf     # Services et sockets
    ├── 10-ssl.conf        # Configuration SSL/TLS
    ├── 15-mailboxes.conf  # Dossiers par défaut
    ├── 20-imap.conf       # Configuration IMAP
    ├── 20-pop3.conf       # Configuration POP3
    └── dovecot-sql.conf.ext # Authentification SQL
```

Répertoires importants

```
/var/mail/          # Boîtes aux lettres (mbox)
/var/spool/mail/    # Alternative mbox
/home/vmail/        # Maildir (recommandé)
/var/log/dovecot/   # Logs Dovecot
```

🔧 Configuration de base

Fichier principal : /etc/dovecot/dovecot.conf

lmtpt = Local Mail Transfer Protocol (pour l'intégration avec Postfix) imap = Internet Message Access Protocol (pour les clients IMAP)

pop3 = Post Office Protocol 3 (pour les clients POP3)

```
# Protocoles activés
protocols = imap pop3 lmtpt

# Écouter sur toutes les interfaces
listen = *, ::

# Fichier de log principal
log_path = /var/log/dovecot/dovecot.log
info_log_path = /var/log/dovecot/info.log
debug_log_path = /var/log/dovecot/debug.log

# Niveau de log
mail_debug = no
auth_debug = no
auth_debug_passwords = no
```

[Revenir au sommaire](#)

Configuration du stockage des emails

Fichier : /etc/dovecot/conf.d/10-mail.conf

Format Maildir (recommandé)

```
# Location des boîtes emails  
mail_location = maildir:/var/mail/vhosts/%d/%n
```

Si vous n'utilisez pas de vmail mais un utilisateur classique alors :

```
# Location des boîtes emails  
mail_location = maildir:/home/%u/Maildir/
```

```
# Utilisateur et groupe pour les fichiers  
mail_uid = vmail  
mail_gid = vmail
```

```
# Priviléges  
first_valid_uid = 5000  
last_valid_uid = 5000  
first_valid_gid = 5000  
last_valid_gid = 5000
```

👤 Création de l'utilisateur vmail

```
# Créer le groupe vmail  
groupadd -g 5000 vmail
```

```
# Créer l'utilisateur vmail  
useradd -g vmail -u 5000 vmail -d /var/mail/vhosts
```

```
# Créer le répertoire et définir les permissions  
mkdir -p /var/mail/vhosts  
chown -R vmail:vmail /var/mail/vhosts  
chmod -R 700 /var/mail/vhosts
```

🔒 Configuration de l'authentification

Fichier : /etc/dovecot/conf.d/10-auth.conf

```
# Désactiver l'authentification en clair sauf sur SSL
disable_plaintext_auth = yes
# Mécanismes d'authentification
auth_mechanisms = plain login
# Inclure les fichiers d'authentification
!include auth-system.conf.ext
# !include auth-sql.conf.ext
# !include auth-ldap.conf.ext
```

🔑 Authentification système

Fichier : /etc/dovecot/conf.d/auth-system.conf.ext

```
# Authentification PAM (utilisateurs système)
passdb {
    driver = pam
}

# Base de données utilisateurs système
userdb {
    driver = passwd
}
```

Note : L'authentification système convient pour de vrais utilisateurs Unix, mais pas pour nos **utilisateurs virtuels** !



🔑 Authentification utilisateurs virtuels

Pour les domaines virtuels (recommandé)

Si vous avez configuré des domaines virtuels avec Postfix (comme dans le module 06), vous avez besoin d'une authentification par fichier.

Créer le fichier d'utilisateurs

```
sudo nano /etc/dovecot/users
```



Contenu :

```
# Format : utilisateur@domaine:{PLAIN}motdepasse
johndoe@andromed.cloud:{PLAIN}MotDePasse123!
janedoe@andromed.cloud:{PLAIN}MotDePasse456!
contact@andromed.cloud:{PLAIN}MotDePasse789!
admin@andromed.cloud:{PLAIN}AdminPass123!
```

⚠ Important : En production, utilisez des hash (SHA512-CRYPT) !

```
# Générer un hash sécurisé  
doveadm pw -s SHA512-CRYPT
```



Créer le fichier de configuration

```
sudo nano /etc/dovecot/conf.d/auth-passwdfile.conf.ext
```



Contenu :

```
passdb {  
    driver = passwd-file  
    args = scheme=PLAIN username_format=%u /etc/dovecot/users  
}  
  
userdb {  
    driver = static  
    args = uid=vmail gid=vmail home=/var/mail/vhosts/%d/%n  
}
```



Activer cette authentification

Dans /etc/dovecot/conf.d/10-auth.conf :

```
# Désactiver l'auth système
#!include auth-system.conf.ext

# Activer l'auth par fichier pour utilisateurs virtuels
!include auth-passwdfile.conf.ext
```



Permissions du fichier

```
sudo chmod 640 /etc/dovecot/users
sudo chown root:dovecot /etc/dovecot/users
```



Redémarrer Dovecot

```
sudo systemctl restart dovecot
```



Tester l'authentification

```
doveadm auth test john Doe@andromed.cloud MotDePasse123!
```



Résultat attendu :

```
passdb: johndoe@andromed.cloud auth succeeded
userdb: johndoe@andromed.cloud
  home   : /var/mail/vhosts/andromed.cloud/johndoe
  uid    : 5000
  gid    : 5000
```

💡 Configuration des ports et services

Fichier : /etc/dovecot/conf.d/10-master.conf

Ports standards

- IMAP : **143** (non chiffré) / **993** (SSL/TLS)
- POP3 : **110** (non chiffré) / **995** (SSL/TLS)
- LMTP : **24** (local uniquement)



Configuration du service IMAP

Ce qu'on va faire ici c'est activer les protocoles IMAP et IMAPS.

```
service imap-login {
    inet_listener imap {
        port = 143
        # Désactiver si vous n'utilisez que SSL
        #port = 0
    }

    inet_listener imaps {
        port = 993
        ssl = yes
    }
}
```



💡 Configuration du service POP3

Ce qu'on va faire ici c'est activer le protocole POP3 et POP3S.

```
service pop3-login {
    inet_listener pop3 {
        port = 110
        # Désactiver si vous n'utilisez que SSL
        #port = 0
    }
    # Activer si vous utilisez SSL
    inet_listener pop3s {
        port = 995
        ssl = yes
    }
}
```



🚀 Configuration du service LMTP

Pour l'intégration avec Postfix

Ce qu'on va faire ici c'est activer le protocole LMTP.

Pourquoi l'activer ? Parce que Postfix va utiliser LMTP pour envoyer les emails à Dovecot.

Nous n'allons pas l'utiliser lors de cette formation , mais il est important de le savoir.

```
service lmtp {
  unix_listener /var/spool/postfix/private/dovecot-lmtp {
    mode = 0600
    user = postfix
    group = postfix
  }
}
```

[Revenir au sommaire](#)

🔒 Configuration de l'authentification SMTP

Pour l'envoi via Postfix

Ce qu'on va faire ici c'est activer l'authentification SMTP via Dovecot. Comme ça on va pouvoir se connecter à Dovecot avec Postfix.

```
service auth {  
    unix_listener /var/spool/postfix/private/auth {  
        mode = 0660  
        user = postfix  
        group = postfix  
    }  
  
    unix_listener auth-userdb {  
        mode = 0600  
        user = vmail  
    }  
}
```

🔒 Configuration SSL/TLS

Fichier : /etc/dovecot/conf.d/10-ssl.conf

```
# Activer SSL (important)
ssl = required

# Certificats SSL (important, si vous utilisez Let's Encrypt => ceux du domaine)
ssl_cert = </etc/letsencrypt/live/mail.example.com/fullchain.pem
ssl_key = </etc/letsencrypt/live/mail.example.com/privkey.pem

# Chaine de certificats
ssl_ca = </etc/letsencrypt/live/mail.example.com/chain.pem
```

```
# Protocoles et chiffrements modernes (2025)
ssl_min_protocol = TLSv1.2
ssl_cipher_list = ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384
ssl_prefer_server_ciphers = yes
```

Bonus :

```
# Paramètres DH (Diffie-Hellman)
ssl_dh = </etc/dovecot/dh.pem
```

🔑 Génération des paramètres DH

```
# Générer des paramètres DH 4096 bits (peut prendre du temps)
openssl dhparam -out /etc/dovecot/dh.pem 4096
```

```
# Permissions
chmod 600 /etc/dovecot/dh.pem
chown root:root /etc/dovecot/dh.pem
```

🔥 Configuration du Firewall

UFW (Ubuntu/Debian)

```
# IMAP
ufw allow 143/tcp comment 'Dovecot IMAP'
ufw allow 993/tcp comment 'Dovecot IMAPS'
```

```
# POP3
ufw allow 110/tcp comment 'Dovecot POP3'
ufw allow 995/tcp comment 'Dovecot POP3S'
```

```
# Recharger le firewall
ufw reload
```

🔥 Configuration Firewalld (CentOS/RHEL)

```
# IMAP
firewall-cmd --permanent --add-service=imap
firewall-cmd --permanent --add-service=imaps
```

```
# POP3
firewall-cmd --permanent --add-service=pop3
firewall-cmd --permanent --add-service=pop3s
```

```
# Recharger
firewall-cmd --reload
```



🔗 Intégration avec Postfix

Configuration de Postfix pour utiliser Dovecot

Fichier /etc/postfix/main.cf

Si vous utilisez lmtp :

```
# Utiliser Dovecot LMTP pour la livraison locale (dans le main.cf de postfix)
mailbox_transport = lmtp:unix:private/dovecot-lmtp
```

A mettre dans le main.cf de postfix :

```
# Authentification SMTP via Dovecot (dans le main.cf de postfix)
smtpd_sasl_type = dovecot
smtpd_sasl_path = private/auth
smtpd_sasl_auth_enable = yes
```



SASL = Simple Authentication and Security Layer

Protocole qui permet l'authentification sécurisée des utilisateurs pour l'envoi d'emails via SMTP. Il évite que n'importe qui puisse utiliser votre serveur comme relais.

```
# Options SASL (dans le main.cf de postfix)
smtpd_sasl_security_options = noanonymous
smtpd_sasl_local_domain = $myhostname
broken_sasl_auth_clients = yes

# Restreindre le relais aux utilisateurs authentifiés
# N'oubliez pas d'activer le paramètre smtpd_sasl_auth_enable = yes dans le fichier main.cf de postfix
smtpd_relay_restrictions =
    permit_mynetworks,
    permit_sasl_authenticated,
    reject_unauth_destination,
    # + les autres restrictions que nous avons vu avant
```

Redémarrage des services

```
# Tester la configuration Dovecot  
doveconf -n
```

```
# Redémarrer Dovecot  
systemctl restart dovecot
```

```
# Redémarrer Postfix  
systemctl restart postfix
```

```
# Vérifier les status  
systemctl status dovecot  
systemctl status postfix
```

Configuration Apple Mail

Étape 1 : Ajouter un compte

1. Ouvrir Mail
2. Mail → Comptes → Ajouter un compte...
3. Sélectionner Autre compte Mail...
4. Cliquer sur Compte Mail

Étape 2 : Informations du compte

Informations à saisir :

- **Nom** : Votre nom complet
- **Adresse e-mail** : votre.email@example.com
- **Mot de passe** : Votre mot de passe

Étape 3 : Configuration manuelle

Si la configuration automatique échoue :

Serveur de réception (IMAP)

- **Type** : IMAP
- **Serveur** : mail.example.com
- **Port** : 993
- **SSL** : Activé
- **Authentification** : Mot de passe



Serveur d'envoi (SMTP)

- **Serveur :** mail.example.com
- **Port :** 587 (ou 465 pour SMTPS)
- **SSL :** Activé
- **Authentification :** Mot de passe
- **Nom d'utilisateur :** votre.email@example.com



Étape 4 : Paramètres avancés

1. Aller dans Préférences → Comptes
2. Sélectionner votre compte
3. Onglet Avancé



Configuration Outlook (Windows)

Étape 1 : Ajouter un compte

1. Ouvrir Outlook
2. Fichier → Ajouter un compte
3. Entrer votre adresse email
4. Cliquer sur **Options avancées**
5. Cocher **Configurer manuellement mon compte**



Étape 2 : Choisir le type de compte

- Sélectionner IMAP



Étape 3 : Configuration IMAP

Courrier entrant

- **Serveur :** mail.example.com
- **Port :** 993
- **Méthode de chiffrement :** SSL/TLS
- **Méthode d'authentification :** Authentification par mot de passe normale



Courrier sortant

- **Serveur :** mail.example.com
- **Port :** 587
- **Méthode de chiffrement :** STARTTLS
- **Méthode d'authentification :** Authentification par mot de passe normale



Informations de connexion

- **Nom d'utilisateur :** votre.email@example.com
- **Mot de passe :** Votre mot de passe



Étape 4 : Paramètres avancés

1. Clic droit sur le compte → **Paramètres du compte**
2. **Plus de paramètres**
3. Onglet **Serveur sortant**

Configuration SMTP :

- Cocher **Mon serveur sortant (SMTP) requiert une authentification**
- Sélectionner **Utiliser les mêmes paramètres que mon serveur de courrier entrant**

Étape 5 : Dossiers

Onglet Avancé :

- Définir le comportement des dossiers supprimés
- Configurer le stockage des éléments envoyés



Configuration Outlook (Mac)

Configuration similaire à Windows

Différences principales :

1. Outlook → Préférences → Comptes
2. Cliquer sur + → Nouveau compte
3. Saisir l'adresse email
4. Cliquer sur **Configurer manuellement**

Les paramètres serveurs sont identiques :

- **IMAP** : mail.example.com:993 (SSL/TLS)
- **SMTP** : mail.example.com:587 (STARTTLS)



Configuration clients mobiles

iOS (iPhone/iPad)

1. Réglages → Mail → Comptes
2. Ajouter un compte → Autre
3. Ajouter un compte Mail

Informations à saisir :

- **Nom** : Votre nom
- **Adresse** : votre.email@example.com
- **Mot de passe** : Votre mot de passe
- **Description** : Nom du compte

Serveur de réception :

- **Nom d'hôte :** mail.example.com
- **Nom d'utilisateur :** votre.email@example.com
- **Mot de passe :** Votre mot de passe

Serveur d'envoi :

- **Nom d'hôte :** mail.example.com
- **Nom d'utilisateur :** votre.email@example.com
- **Mot de passe :** Votre mot de passe

Android (Gmail App)

1. Ouvrir Gmail
2. Menu → Paramètres
3. Ajouter un compte → Autre



Configuration automatique ou manuelle :

- Type : **IMAP personnel**
- Serveur : mail.example.com
- Port : **993**
- Sécurité : **SSL/TLS**



🔍 Tests et vérification

Tester l'authentification

```
# Tester l'auth avec doveadm  
doveadm auth test johndoe@andromed.cloud
```

Tester l'user

```
# Tester l'user avec doveadm  
doveadm user johndoe@andromed.cloud
```

Vérifier les ports ouverts

Vérifier les ports en écoute
`netstat -tlnp | grep dovecot`



```
# Alternative avec ss  
ss -tlnp | grep dovecot
```



Résultat attendu

```
tcp 0 0 0.0.0.0:143 0.0.0.0:* LISTEN 1234/dovecot
tcp 0 0 0.0.0.0:993 0.0.0.0:* LISTEN 1234/dovecot
tcp 0 0 0.0.0.0:110 0.0.0.0:* LISTEN 1234/dovecot
tcp 0 0 0.0.0.0:995 0.0.0.0:* LISTEN 1234/dovecot
```

🧪 Tester IMAP manuellement

```
# Connexion IMAP non chiffrée (test uniquement)
telnet mail.andromed.cloud 143
```



Commandes à tester :

```
a001 LOGIN johndoe@andromed.cloud password  
a002 LIST "" "*"  
a003 SELECT INBOX  
a004 LOGOUT
```



🧪 Tester IMAPS avec OpenSSL

```
# Connexion IMAPS chiffrée  
openssl s_client -connect mail.andromed.cloud:993
```



Puis tester l'authentification :

```
a001 LOGIN utilisateur@example.com password  
a002 LIST "" "*"  
a003 LOGOUT
```



🧪 Tester POP3 manuellement

```
# Connexion POP3 non chiffrée  
telnet mail.andromed.cloud 110
```



Commandes à tester :

```
USER johndoe@andromed.cloud
PASS password
STAT
LIST
QUIT
```



🧪 Tester POP3S avec OpenSSL

```
# Connexion POP3S chiffrée  
openssl s_client -connect mail.andromed.cloud:995
```



Surveillance et logs

Fichiers de logs

```
# Log principal  
tail -f /var/log/dovecot/dovecot.log
```

```
# Log d'information  
tail -f /var/log/dovecot/info.log
```

```
# Log de debug (si activé)  
tail -f /var/log/dovecot/debug.log
```

Activer les logs de debug

Fichier /etc/dovecot/dovecot.conf

```
# Activer temporairement pour déboguer
mail_debug = yes
auth_debug = yes
auth_debug_passwords = yes
auth_verbose = yes
```

⚠ Attention : Désactiver après débogage (les mots de passe apparaissent en clair)

🔧 Commandes utiles doveadm

Gestion des utilisateurs

```
# Lister les boîtes aux lettres  
doveadm mailbox list -u johndoe@andromed.cloud  
  
# Vérifier le quota d'un utilisateur  
doveadm quota get -u johndoe@andromed.cloud  
  
# Forcer une réindexation  
doveadm index -u johndoe@andromed.cloud INBOX
```

```
# Purger les emails marqués comme supprimés  
doveadm expunge -u johndoe@andromed.cloud mailbox INBOX all
```

Recharger la configuration

```
# Recharger sans couper les connexions  
doveadm reload
```

```
# Lister les connexions actives  
doveadm who
```

```
# Déconnecter un utilisateur  
doveadm kick johndoe@andromed.cloud
```



Troubleshooting courant

Problème : Impossible de se connecter

Vérifications :

1. Vérifier que Dovecot est démarré

```
systemctl status dovecot
```

2. Vérifier les ports ouverts

```
ss -tlnp | grep dovecot
```

3. Vérifier le firewall

```
# UFW  
ufw status  
  
# Firewalld  
firewall-cmd --list-all
```

4. Vérifier les certificats SSL

```
openssl s_client -connect mail.example.com:993 -showcerts
```

Problème : Erreur d'authentification

Vérifications :

```
# Tester l'authentification  
doveadm auth test utilisateur@example.com
```

```
# Vérifier les logs  
tail -100 /var/log/dovecot/dovecot.log | grep auth
```

```
# Vérifier les permissions  
ls -la /var/mail/vhosts/
```

Problème : Certificat SSL invalide

Vérifications :

1. Vérifier la validité du certificat

```
openssl x509 -in /etc/letsencrypt/live/mail.example.com/cert.pem -text -noout
```

2. Renouveler avec Let's Encrypt

```
certbot renew  
systemctl restart dovecot
```

3. Vérifier la configuration SSL dans Dovecot

```
doveconf -n | grep ssl
```

Problème : Emails non livrés

Vérifications :

```
# Vérifier l'intégration avec Postfix  
postconf | grep mailbox_transport
```

Si vous utilisez lmtp :

```
# Vérifier le socket LMTP  
ls -la /var/spool/postfix/private/dovecot-lmtp  
  
# Tester LMTP manuellement  
doveadm exec lmtp
```

Optimisations performances

Configuration pour serveur haute charge

Fichier /etc/dovecot/dovecot.conf

```
# Augmenter les limites de connexions
default_process_limit = 1000
default_client_limit = 10000
```

```
# Nombre de processus de login
service imap-login {
    process_min_avail = 10
    service_count = 0
}
```

```
# Cache d'authentification
auth_cache_size = 100M
auth_cache_ttl = 1 hour
auth_cache_negative_ttl = 1 hour
```

E Configuration Sieve (filtres)

Activer Sieve (si vous utilisez lmtp)

Fichier /etc/dovecot/conf.d/20-lmtp.conf

```
protocol lmtp {
    mail_plugins = $mail_plugins sieve
}
```

Fichier /etc/dovecot/conf.d/90-sieve.conf

```
plugin {
    sieve = file:~/sieve;active=~/dovecot.sieve
    sieve_dir = ~/sieve
    sieve_global_dir = /var/lib/dovecot/sieve/global/
}
```

Exemple de règle Sieve

```
require ["fileinto", "mailbox"];  
  
# Déplacer les spams dans Junk  
if header :contains "X-Spam-Flag" "YES" {  
    fileinto :create "Junk";  
    stop;  
}
```

```
# Filtrer par expéditeur
if address :is "from" "newsletter@example.com" {
    fileinto :create "Newsletters";
    stop;
}
```



Statistiques avec doveadm

Statistiques générales
`doveadm stats dump`

Statistiques par utilisateur
`doveadm stats dump user`

Statistiques des sessions
`doveadm stats dump session`



🔒 Sécurité supplémentaire

Limiter les tentatives de connexion

Fichier /etc/dovecot/dovecot.conf

```
# Protection contre les attaques par force brute
auth_policy_server_url = http://localhost:9090/
auth_policy_server_api_header = Authorization: Bearer secret
auth_policy_hash_mech = sha256
```

Fail2ban pour Dovecot

Fichier /etc/fail2ban/jail.local

```
[dovecot]
enabled = true
port = imap,imaps,pop3,pop3s
filter = dovecot
logpath = /var/log/dovecot/dovecot.log
maxretry = 5
bantime = 3600
```



🎯 Checklist de configuration

Avant la mise en production

- Dovecot installé et démarré
- Ports 143, 993, 110, 995 ouverts dans le firewall
- Certificats SSL valides et configurés
- SSL forcé (disable_plaintext_auth = yes)
- Intégration avec Postfix fonctionnelle
- Authentification testée

- Tests manuels IMAP/POP3 réussis
- Configuration client email validée (Apple Mail, Outlook)
- Logs activés et surveillés
- Sauvegardes configurées
- Quotas définis (si nécessaire)
- Fail2ban activé
- DNS MX, SPF, DKIM, DMARC configurés



Ressources complémentaires

Documentation officielle :

- [Dovecot Wiki](#)
- [Dovecot Configuration](#)
- [Dovecot + Postfix](#)

Outils de test :

- [IMAP Test Tool](#)
- [SSL Labs](#) - Test SSL/TLS



Points clés à retenir

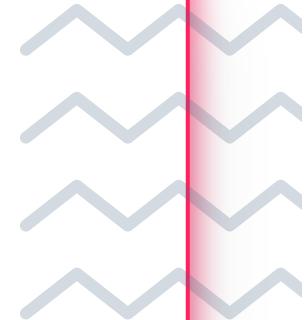
1. Dovecot complète Postfix : Postfix envoie/reçoit, Dovecot stocke/distribue
2. IMAP vs POP3 : IMAP garde les emails sur le serveur (recommandé)
3. Ports : 993 (IMAPS) et 587 (SMTP+STARTTLS) en production
4. SSL/TLS obligatoire : Jamais de connexions non chiffrées en production
5. Maildir > Mbox : Format moderne et performant

6. **Authentification** : SASL via Dovecot pour Postfix
7. **LMTP** : Protocole de livraison locale recommandé
8. **Logs** : Toujours surveiller /var/log/dovecot/
9. **Sieve** : Filtres côté serveur puissants
10. **Sécurité** : Fail2ban + certificats + chiffrement



Exercices pratiques

Configuration Dovecot





Exercice 1 : Installation de base

Objectif : Installer et configurer Dovecot

Durée : 20 minutes

Tâches :

1. Installer Dovecot avec IMAP et POP3
2. Créer l'utilisateur vmail (UID/GID 5000)
3. Configurer le répertoire /var/mail/vhosts
4. Définir mail_location en Maildir dans le fichier /etc/dovecot/conf.d/10-mail.conf

5. Activer les protocoles IMAP et POP3
6. Configurer les logs dans /var/log/dovecot/
7. Démarrer et activer Dovecot au boot
8. Vérifier que les ports 143, 993, 110, 995 sont en écoute



Exercice 2 : Configuration SSL/TLS

Objectif : Sécuriser Dovecot avec SSL/TLS

Durée : 30 minutes

Tâches :

1. Obtenir un certificat Let's Encrypt pour mail.andromed.cloud
2. Configurer les chemins des certificats dans /etc/dovecot/conf.d/10-ssl.conf
3. Forcer SSL avec ssl = required
4. Définir ssl_min_protocol = TLSv1.2

Optionnel :

5. Générer les paramètres DH 4096 bits



Exercice 3 : Intégration avec Postfix

Objectif : Connecter Dovecot et Postfix

Durée : 25 minutes

Tâches :

2. Configurer le socket d'authentification pour Postfix
3. Modifier `main.cf` de Postfix :
 - `smtpd_sasl_path = private/auth`

4. Activer l'authentification SMTP
5. Redémarrer Dovecot et Postfix
6. Envoyer un email de test
7. Vérifier la livraison dans le Maildir





Exercice 4 : Configuration client

Objectif : Configurer un client email

Durée : 15 minutes

Tâches :

1. Configurer Apple Mail ou Outlook avec :
 - IMAP : mail.example.com:993 (SSL/TLS)
 - SMTP : mail.example.com:587 (STARTTLS)
2. Tester l'envoi d'un email
3. Tester la réception d'un email

4. Vérifier la création des dossiers IMAP
5. Configurer les dossiers spéciaux (Envoyés, Brouillons)
6. Tester la synchronisation IMAP



Exercice 5 : Troubleshooting

Objectif : Diagnostiquer et résoudre des problèmes

Durée : 20 minutes

Scénarios à résoudre :

1. Erreur : "Authentication failed"

- Vérifier les logs
- Tester avec `doveadm auth test`
- Vérifier les permissions

2. Erreur : "Connection refused" sur le port 993

- Vérifier que Dovecot est démarré
- Vérifier le firewall
- Vérifier la configuration SSL

3. Erreur : Emails non livrés dans le Maildir

- Vérifier le socket LMTP
- Vérifier la configuration Postfix
- Consulter les logs de Postfix et Dovecot



Exercice 6 : Configuration avancée

Objectif : Configurer des fonctionnalités avancées

Durée : 30 minutes

Tâches :

1. Configurer les quotas (1GB par utilisateur)
2. Activer Sieve pour les filtres
3. Créer une règle Sieve pour filtrer le spam
4. Configurer Fail2ban pour Dovecot

5. Activer le cache d'authentification
6. Optimiser les paramètres de performance
7. Configurer les statistiques avec `doveadm stats`

✓ Solutions

Exercice 1 : Installation

```
# Installation  
apt update  
apt install -y dovecot-core dovecot-imapd dovecot-pop3d
```

Le compte vmail est utilisé pour gérer les boîtes aux lettres des utilisateurs virtuels. Il permet à Dovecot de stocker tous les emails sous un seul utilisateur système dédié, simplifiant la gestion des permissions et améliorant la sécurité, sans que chaque adresse email ait un compte système correspondant.

```
# Utilisateur vmail
groupadd -g 5000 vmail
useradd -g vmail -u 5000 vmail -d /var/mail/vhosts
mkdir -p /var/mail/vhosts
chown -R vmail:vmail /var/mail/vhosts
chmod -R 700 /var/mail/vhosts
```



```
# Configuration dans /etc/dovecot/conf.d/10-mail.conf
mail_location = maildir:/var/mail/vhosts/%d/%n
mail_uid = vmail
mail_gid = vmail
first_valid_uid = 5000
```

```
# Démarrage  
systemctl start dovecot  
systemctl enable dovecot  
systemctl status dovecot
```

[Revenir au sommaire](#)

```
# Vérification  
ss -tlnp | grep dovecot
```



Exercice 2 : SSL/TLS

```
# Certificat Let's Encrypt  
certbot certonly --standalone -d mail.andromed.cloud
```



```
# Configuration /etc/dovecot/conf.d/10-ssl.conf
ssl = required
ssl_cert = </etc/letsencrypt/live/mail.andromed.cloud/fullchain.pem
ssl_key = </etc/letsencrypt/live/mail.andromed.cloud/privkey.pem
ssl_min_protocol = TLSv1.2
```

```
# Paramètres DH  
openssl dhparam -out /etc/dovecot/dh.pem 4096
```



```
# Désactiver ports non chiffrés dans /etc/dovecot/conf.d/10-master.conf
service imap-login {
    inet_listener imap {
        port = 0
    }
}
service pop3-login {
    inet_listener pop3 {
        port = 0
    }
}
```



```
# Redémarrer  
systemctl restart dovecot
```



```
# Test  
openssl s_client -connect mail.andromed.cloud:993
```



Exercice 3 : Intégration Postfix

```
service auth {  
    unix_listener /var/spool/postfix/private/auth {  
        mode = 0660  
        user = postfix  
        group = postfix  
    }  
}
```



A mettre dans le main.cf de postfix :

```
# /etc/postfix/main.cf
mailbox_transport = lmtp:unix:private/dovecot-lmtp
smtpd_sasl_type = dovecot
smtpd_sasl_path = private/auth
smtpd_sasl_auth_enable = yes
```



```
# Redémarrer les services  
systemctl restart dovecot  
systemctl restart postfix
```



Exercice 4 : Configuration client

Apple Mail :

- Ajouter un compte → Autre compte Mail
- IMAP : mail.example.com:993 (SSL)
- SMTP : mail.example.com:587 (TLS)
- Identifiant complet : user@example.com

Outlook :

- Fichier → Ajouter un compte → Configuration manuelle
- IMAP : mail.example.com:993 (SSL/TLS)
- SMTP : mail.example.com:587 (STARTTLS)
- Identifiant complet : user@example.com



Exercice 5 : Troubleshooting

Authentication failed :

```
# Vérifier l'authentification  
doveadm auth test user@example.com
```

```
# Logs  
tail -f /var/log/dovecot/dovecot.log
```



```
# Permissions  
ls -la /var/mail/vhosts/
```



Connection refused :

```
# Dovecot actif ?  
systemctl status dovecot
```



```
# Port ouvert ?  
ss -tlnp | grep 993
```



```
# Firewall ?  
ufw status  
ufw allow 993/tcp
```



Emails non livrés :

```
# Socket LMTP existe ?  
ls -la /var/spool/postfix/private/dovecot-lmtp
```

```
# Configuration Postfix  
postconf mailbox_transport
```



```
# Logs
tail -f /var/log/mail.log
tail -f /var/log/dovecot/dovecot.log
```



Exercice 6 : Configuration avancée

```
# /etc/dovecot/conf.d/90-quota.conf
plugin {
    quota = maildir:User quota
    quota_rule = *:storage=1GB
}
```



```
# Activer quota dans /etc/dovecot/conf.d/10-mail.conf
mail_plugins = $mail_plugins quota
```



```
# Sieve dans /etc/dovecot/conf.d/20-lmtp.conf
protocol lmtp {
    mail_plugins = $mail_plugins sieve
}
```



```
# Fail2ban  
apt install fail2ban
```



Fichier /etc/fail2ban/jail.local :

```
[dovecot]
enabled = true
port = imap,imaps,pop3,pop3s
filter = dovecot
logpath = /var/log/dovecot/dovecot.log
maxretry = 5
```

```
# Redémarrer  
systemctl restart fail2ban  
systemctl restart dovecot
```

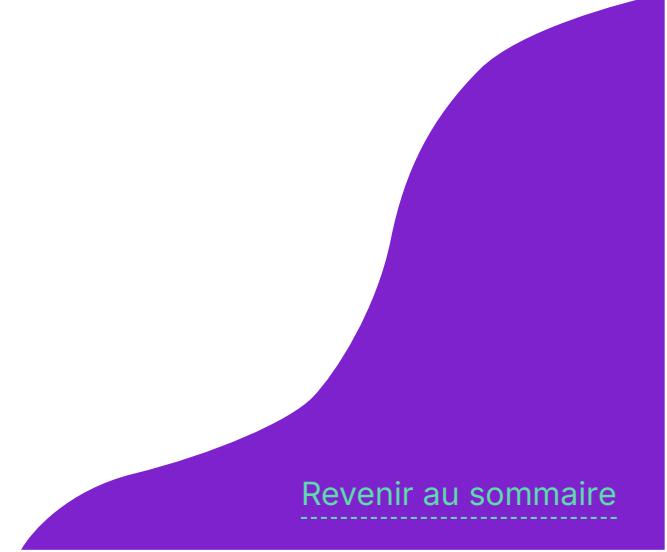




Fin du module Dovecot

Vous savez maintenant :

- Installer et configurer Dovecot
- Intégrer Dovecot avec Postfix
- Sécuriser avec SSL/TLS
- Configurer les clients email
- Diagnostiquer les problèmes
- Optimiser les performances



[Revenir au sommaire](#)



QCM Dovecot

Validation des connaissances

Question 1 : Rôle de Dovecot

Quel est le rôle principal de Dovecot dans un serveur de messagerie ?

- A) Envoyer des emails vers d'autres serveurs
- B) Recevoir des emails depuis Internet
- C) Permettre aux utilisateurs de récupérer leurs emails (IMAP/POP3)
- D) Filtrer les spams

Question 2 : Ports standards

Quels sont les ports standards pour IMAPS et SMTPS ?

- A) 143 et 25
- B) 993 et 465
- C) 995 et 587
- D) 110 et 25

Question 3 : Format de stockage

Quel format de stockage des emails est recommandé avec Dovecot en 2025 ?

- A) mbox
- B) Maildir
- C) dbox
- D) mdbox

Question 4 : Authentification SMTP

Comment Postfix authentifie-t-il les utilisateurs pour l'envoi d'emails via Dovecot ?

- A) Via PAM uniquement
- B) Via le socket auth de Dovecot (SASL)
- C) Via une base de données MySQL directe
- D) Via des fichiers passwd



Question 5 : Protocole LMTP

Pourquoi utiliser LMTP au lieu de LDA pour la livraison locale ?

- A) LMTP est plus rapide
- B) LMTP supporte la livraison simultanée à plusieurs destinataires
- C) LMTP ne nécessite pas de priviléges élevés
- D) Toutes les réponses ci-dessus



Question 6 : SSL/TLS

Quelle directive Dovecot force l'utilisation de SSL/TLS ?

- A) ssl = yes
- B) ssl = required
- C) ssl_force = yes
- D) disable_plaintext_auth = yes

Question 7 : Utilisateur vmail

Pourquoi créer un utilisateur système 'vmail' pour Dovecot ?

- A) C'est obligatoire pour compiler Dovecot
- B) Pour séparer les privilèges et sécuriser les boîtes emails
- C) Pour la compatibilité avec Postfix uniquement
- D) Ce n'est pas nécessaire

Question 8 : Configuration client

Quel paramètre SMTP doit-on utiliser pour l'envoi d'emails en 2025 ?

- A) Port 25 sans authentification
- B) Port 465 avec SSL/TLS
- C) Port 587 avec STARTTLS
- D) B et C sont corrects



Question 9 : IMAP vs POP3

Quelle est la principale différence entre IMAP et POP3 ?

- A) IMAP est plus sécurisé que POP3
- B) IMAP garde les emails sur le serveur, POP3 les télécharge
- C) POP3 est plus rapide qu'IMAP
- D) IMAP ne supporte pas SSL/TLS

Question 10 : Sieve

Qu'est-ce que Sieve dans Dovecot ?

- A) Un protocole de synchronisation
- B) Un langage de filtrage des emails côté serveur
- C) Un outil de compression des boîtes emails
- D) Un système de sauvegarde

Question 11 : Logs Dovecot

Où se trouvent les logs de Dovecot par défaut ?

- A) /var/log/mail.log
- B) /var/log/dovecot/dovecot.log
- C) /var/log/syslog
- D) /var/log/messages



Question 12 : Commande dovecadm

Que fait la commande `doveadm auth test user@example.com ?`

- A) Crée un nouvel utilisateur
- B) Teste l'authentification d'un utilisateur
- C) Supprime un utilisateur
- D) Change le mot de passe

Question 13 : Certificats SSL

Quelle commande permet de tester une connexion IMAPS ?

- A) telnet mail.example.com 993
- B) openssl s_client -connect mail.example.com:993
- C) curl https://mail.example.com:993
- D) nc mail.example.com 993



Question 14 : Quotas

Comment définir un quota de 1GB par utilisateur dans Dovecot ?

- A) quota = 1GB
- B) quota_rule = *:storage=1GB
- C) user_quota = 1024M
- D) mailbox_size_limit = 1GB

Question 15 : Paramètres DH

Pourquoi générer des paramètres DH pour Dovecot ?

- A) Pour améliorer la sécurité SSL/TLS (Perfect Forward Secrecy)
- B) Pour accélérer les connexions
- C) Pour la compatibilité avec Outlook
- D) Ce n'est plus nécessaire en 2025

Question 16 : Firewall

Quelles commandes UFW sont nécessaires pour Dovecot ?

- A) ufw allow 25,587
- B) ufw allow 143,993,110,995
- C) ufw allow 80,443
- D) ufw allow 3306

Question 17 : Intégration Postfix

Quelle directive Postfix définit l'utilisation de Dovecot LMTP ?

- A) mailbox_command = dovecot-lmtp
- B) virtual_transport = lmtp:unix:private/dovecot-lmtp
- C) mailbox_transport = lmtp:unix:private/dovecot-lmtp
- D) local_transport = dovecot

Question 18 : Fail2ban

Pourquoi configurer Fail2ban avec Dovecot ?

- A) Pour améliorer les performances
- B) Pour protéger contre les attaques par force brute
- C) Pour compresser les logs
- D) Pour créer des sauvegardes automatiques

Question 19 : Format mail_location

Que signifie `mail_location = maildir:/var/mail/vhosts/%d/%n` ?

- A) %d = nom d'utilisateur, %n = domaine
- B) %d = domaine, %n = nom d'utilisateur (avant @)
- C) %d = date, %n = numéro
- D) %d = répertoire, %n = nom complet

Question 20 : Debug mode

Comment activer temporairement les logs de debug dans Dovecot ?

- A) debug = yes dans dovecot.conf
- B) mail_debug = yes et auth_debug = yes
- C) loglevel = debug
- D) dovecot --debug



Réponses - QCM Dovecot (1/2)

Question 1 : Réponse C - Dovecot est un serveur IMAP/POP3 qui permet aux clients email de récupérer leurs messages.

Question 2 : Réponse B - 993 : IMAPS / 465 : SMTPS / 143 : IMAP / 587 : SMTP STARTTLS / 110 : POP3 / 995 : POP3S

Question 3 : Réponse B - **Maildir** : un fichier par email, concurrent-safe, performant, compatible.

Question 4 : Réponse B - Via socket Unix : `smtpd_sasl_type = dovecot` et `smtpd_sasl_path = private/auth`

Question 5 : Réponse D - LMTP = plus rapide, livraison groupée, pas de privilèges root, meilleure gestion erreurs.

Question 6 : Réponse B - `ssl = required` force SSL/TLS sur toutes les connexions.

Question 7 : Réponse B - Séparation des privilèges, sécurité, UID/GID fixes, isolation des boîtes.

Question 8 : Réponse D - Port 587 + STARTTLS (recommandé) et Port 465 + SSL/TLS sont valides en 2025.

Question 9 : Réponse B - IMAP : sync, emails restent sur serveur. POP3 : téléchargement, emails supprimés.

Question 10 : Réponse B - **Sieve** : langage de filtrage côté serveur (tri, spam, réponses auto).

Réponses - QCM Dovecot (2/2)

Question 11 : Réponse B - `/var/log/dovecot/dovecot.log` , `/var/log/dovecot/info.log` , `/var/log/dovecot/debug.log`

Question 12 : Réponse B - `doveadm auth test` teste l'authentification sans client email.

Question 13 : Réponse B - `openssl s_client -connect mail.example.com:993` teste SSL/TLS + certificat.

Question 14 : Réponse B - Dans `90-quota.conf` : `quota_rule = *:storage=1GB`

Question 15 : Réponse A - Paramètres DH = Perfect Forward Secrecy (PFS), 4096 bits recommandé.

Question 16 : Réponse B - Ports IMAP/IMAPS/POP3/POP3S : 143, 993, 110, 995.

Question 17 : Réponse C - `mailbox_transport = lmtp:unix:private/dovecot-lmtp` dans `main.cf`

Question 18 : Réponse B - Protection contre force brute et tentatives de connexion échouées.

Question 19 : Réponse B - `%d` = domaine, `%n` = nom local. Ex : `user@ex.com` → `/var/mail/vhosts/ex.com/user/`

Question 20 : Réponse B - `mail_debug = yes` , `auth_debug = yes` , `auth_debug_passwords = yes` (⚠ désactiver après)

Scoring

Notation :

- **18-20 bonnes réponses** : Excellent ! 🏆 Vous maîtrisez parfaitement Dovecot
- **15-17 bonnes réponses** : Très bien ! 🎯 Solides connaissances, quelques révisions conseillées
- **12-14 bonnes réponses** : Bien 👍 Bonnes bases, revoir certains points avancés
- **Moins de 12** : À revoir 📚 Relire le module et refaire les exercices

🎯 Points clés à retenir

1. **Dovecot** = IMAP/POP3 (récupération des emails)
2. **Postfix** = SMTP (envoi/réception)
3. **LMTP** : Protocole de livraison local recommandé
4. **Maildir** : Format de stockage moderne
5. **SSL/TLS obligatoire** en production
6. **Ports** : 993 (IMAPS), 587 (Submission)
7. **vmail** : Utilisateur dédié pour la sécurité
8. **SASL** : Authentification Postfix via Dovecot
9. **Sieve** : Filtres côté serveur
10. **Fail2ban** : Protection contre les attaques



Pour aller plus loin

Sujets avancés :

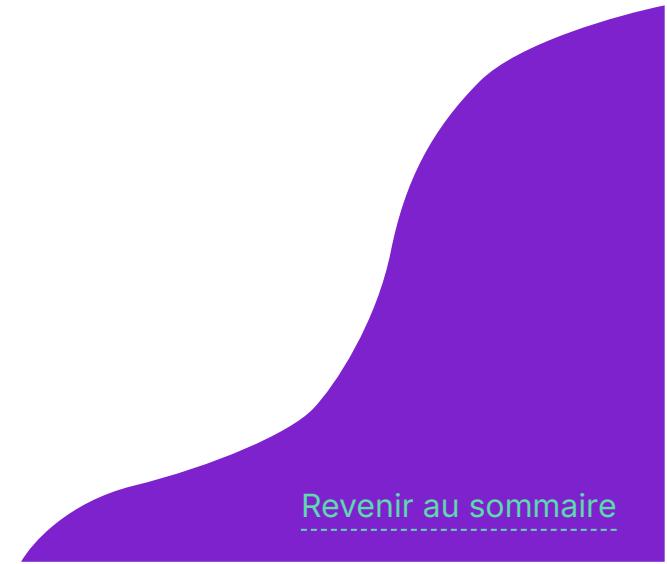
- Haute disponibilité (réPLICATION Dovecot)
- Clustering et load balancing
- Authentification LDAP/Active Directory
- Antispam côté serveur (Rspamd + Sieve)
- Monitoring avancé (Prometheus, Grafana)

🎓 Félicitations !

Vous avez terminé le module Dovecot !

Vous savez maintenant :

- Installer et configurer Dovecot
- Intégrer avec Postfix via LMTP et SASL
- Sécuriser avec SSL/TLS modernes
- Configurer les clients email
- Diagnostiquer et résoudre les problèmes
- Optimiser les performances



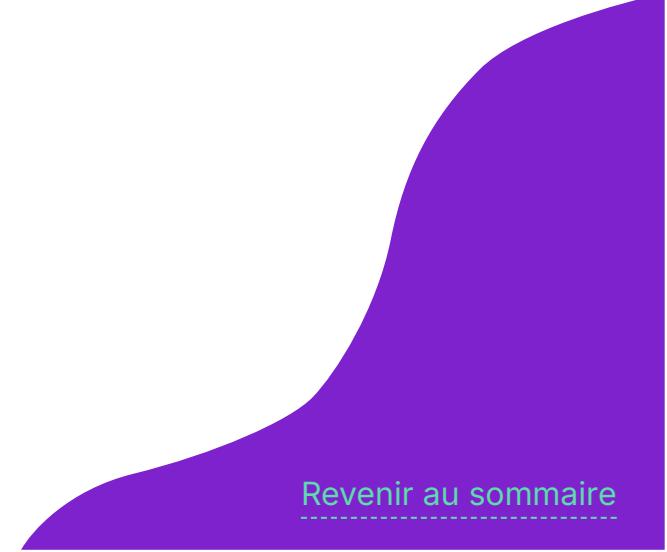
Revenir au sommaire



Prochaines étapes

Pour un serveur de messagerie complet :

1. Postfix configuré et sécurisé
2. Dovecot installé et intégré
3. SPF, DKIM, DMARC configurés
4. Antispam (Rspamd/SpamAssassin)
5. Monitoring (logs, alertes)
6. Sauvegardes automatiques
7. Tests de délivrabilité



[Revenir au sommaire](#)

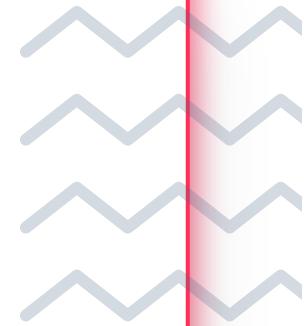


Module Dovecot terminé !



Bravo ! Vous êtes prêt à déployer un serveur de messagerie complet.

[Retour au sommaire →](#)



Protection Anti-Spam



Défendre votre serveur contre les indésirables

Introduction au spam

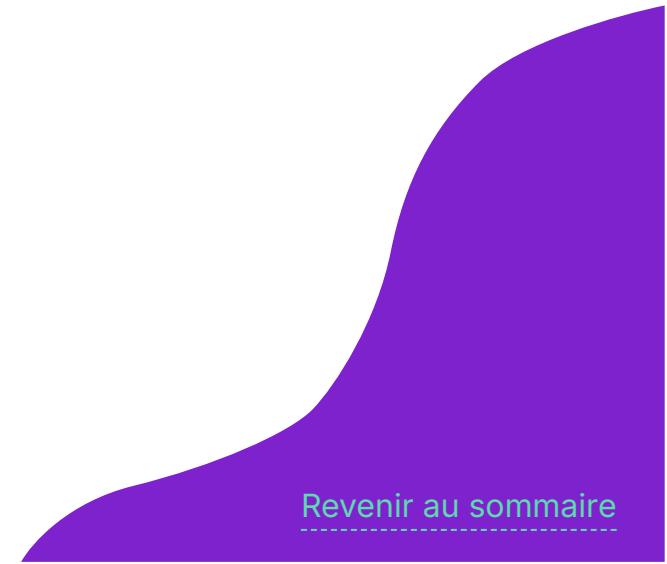
Le spam représente **plus de 50%** du trafic email mondial en 2025 !

Sans protection, votre serveur sera rapidement submergé.

Les types de spam

Spam publicitaire

- Produits douteux
- Arnaques
- Viagra, casinos, etc.



Revenir au sommaire

Phishing

- Usurpation d'identité
- Vol de données bancaires
- Fausses factures

Malware

- Pièces jointes infectées
- Liens malveillants
- Ransomware

Backscatter

- Bounces de spam
- Votre domaine usurpé
- Retour d'emails que vous n'avez pas envoyés

Stratégie de défense en couches

Pensez à un château fort :

1. **Couche 1** : Restrictions réseau (RBL, rate limiting)
2. **Couche 2** : Restrictions SMTP (HELO, sender, recipient, client)

 **Note** : Les protections avancées (Greylisting, SpamAssassin, Rspamd) seront vues dans la formation **Perfectionnement**.

Couche 1 : Restrictions réseau

DNS Blacklists (RBL)

Les RBL sont des listes d'IPs connues pour envoyer du spam.

🔍 Comment ça marche ?

Quand un serveur se connecte à vous :

1. Postfix extrait son IP : 1.2.3.4
2. Inverse l'IP : 4.3.2.1
3. Query DNS : 4.3.2.1.zen.spamhaus.org
4. Si réponse = IP est blacklistée → Rejet

RBL recommandé pour débuter

Spamhaus ZEN (le plus utilisé et fiable) :

- Domaine : zen.spamhaus.org
- Combine plusieurs listes Spamhaus
- Très fiable, peu de faux positifs
- **Le meilleur choix pour une configuration de base**

⚙️ Configuration dans main.cf

```
# Vérification RBL avec Spamhaus ZEN
smtpd_recipient_restrictions =
    permit_mynetworks,
    permit_sasl_authenticated,
    reject_unauth_destination,
    reject_rbl_client zen.spamhaus.org
```



⚠ Attention aux faux positifs !

Les RBL peuvent parfois blacklister des IPs légitimes.

Solution : Whitelister les IPs/domaines de confiance si nécessaire

✓ Tester si une IP est blacklistée

```
# Tester manuellement  
dig 4.3.2.1.zen.spamhaus.org  
  
# Ou avec un outil en ligne  
# https://mxtoolbox.com/blacklists.aspx
```



Rate Limiting

Limiter le nombre de connexions par IP pour empêcher le flood.

⚙️ Configuration avec Anvil

```
# Limite de connexions simultanées par IP  
smtpd_client_connection_count_limit = 10  
  
# Limite de taux (connexions par période)  
smtpd_client_connection_rate_limit = 30  
  
# Durée de la période (secondes)  
anvil_rate_time_unit = 60s
```

Exemple : Avec ces paramètres, une IP peut avoir maximum 10 connexions simultanées et 30 nouvelles connexions par minute. Au-delà → Rejet temporaire

✉️ Messages par connexion

```
# Limite de messages par connexion  
smtpd_client_message_rate_limit = 100
```



Couche 2 : Restrictions SMTP

Vérifications HELO/EHLO

Le HELO est la première chose que dit un serveur SMTP.

Beaucoup de spameurs ont un HELO invalide ou suspect.



🚫 Rejeter HELO invalides

```
smtpd_helo_required = yes  
  
smtpd_helo_restrictions =  
    permit_mynetworks,  
    permit_sasl_authenticated,  
    reject_invalid_helo_hostname,  
    reject_non_fqdn_helo_hostname,  
    reject_unknown_helo_hostname
```



Explications :

- `reject_invalid_helo_hostname` : HELO syntaxiquement incorrect
- `reject_non_fqdn_helo_hostname` : HELO pas en FQDN (ex: "localhost")
- `reject_unknown_helo_hostname` : HELO dont le domaine n'existe pas en DNS

Rejeter HELO se faisant passer pour vous

```
smtpd_helo_restrictions =  
    permit_mynetworks,  
    permit_sasl_authenticated,  
    reject_invalid_helo_hostname,  
    reject_non_fqdn_helo_hostname,  
    check_helo_access hash:/etc/postfix/helo_access
```

Fichier /etc/postfix/helo_access :

```
# Rejeter les serveurs qui prétendent être nous  
mail.example.com      REJECT You are not me!  
example.com           REJECT You are not me!  
192.168.1.10          REJECT You are not my IP!
```

```
sudo postmap /etc/postfix/helo_access  
sudo systemctl reload postfix
```

Vérifications Sender

Vérifier que l'expéditeur est valide.

🚫 Restrictions sender

```
smtpd_sender_restrictions =  
    permit_mynetworks,  
    permit_sasl_authenticated,  
    reject_non_fqdn_sender,  
    reject_unknown_sender_domain
```



Explications :

- `reject_non_fqdn_sender` : Expéditeur pas en FQDN (ex: "user")
- `reject_unknown_sender_domain` : Domaine expéditeur n'existe pas en DNS



Blacklist/Whitelist manuelle

Fichier /etc/postfix/sender_access :

```
# Blacklist
spammer@spam.com      REJECT
@spam-domain.com      REJECT

# Whitelist
trusted@partner.com   OK
@trusted-partner.com  OK
```



```
smtpd_sender_restrictions =  
    check_sender_access hash:/etc/postfix/sender_access,  
    permit_mynetworks,  
    permit_sasl_authenticated,  
    reject_non_fqdn_sender,  
    reject_unknown_sender_domain
```



```
sudo postmap /etc/postfix/sender_access  
sudo systemctl reload postfix
```

Vérifications Recipient

S'assurer que le destinataire est légitime.

🚫 Restrictions recipient

```
smtptd_recipient_restrictions =  
    permit_mynetworks,  
    permit_sasl_authenticated,  
    reject_unauth_destination,  
    reject_non_fqdn_recipient,  
    reject_unknown_recipient_domain,  
    reject_rbl_client zen.spamhaus.org
```



CRUCIAL : `reject_unauth_destination` empêche votre serveur d'être un open relay !

Vérifications Client

Restrictions basées sur l'IP/hostname du client.

🚫 Restrictions client

```
smtpd_client_restrictions =  
    permit_mynetworks,  
    permit_sasl_authenticated,  
    reject_unknown_client_hostname,  
    check_client_access hash:/etc/postfix/client_access
```

Fichier /etc/postfix/client_access :

```
# Blacklist IP
1.2.3.4          REJECT  Spammer IP
10.0.0.0/8        REJECT  Private network

# Whitelist
trusted.partner.com    OK
203.0.113.0/24        OK
```

```
sudo postmap /etc/postfix/client_access  
sudo systemctl reload postfix
```



Pour aller plus loin

Vous pouvez aller plus loin avec les protections avancées suivantes :

- **Greylisting (Postgrey)** : Retarder temporairement les emails inconnus
- **SpamAssassin** : Filtrage de contenu avec score bayésien
- **Rspamd** : Solution moderne et performante
- **Amavis** : Intégration complète anti-virus + anti-spam

Configuration complète anti-spam

Fichier `/etc/postfix/main.cf`

Ajouter ces paramètres pour une protection de base efficace :



```
# =====
# PROTECTION ANTI-SPAM - CONFIGURATION BASE
# =====

# === RESTRICTIONS CLIENT ===
smtpd_client_restrictions =
    permit_mynetworks,
    permit_sasl_authenticated,
    reject_unknown_client_hostname,
    check_client_access hash:/etc/postfix/client_access

# === RESTRICTIONS HELO ===
smtpd_helo_required = yes
smtpd_helo_restrictions =
    permit_mynetworks,
    permit_sasl_authenticated,
    reject_invalid_helo_hostname,
    reject_non_fqdn_helo_hostname,
    reject_unknown_helo_hostname

# === RESTRICTIONS SENDER ===
smtpd_sender_restrictions =
    permit_mynetworks,
    permit_sasl_authenticated,
    reject_non_fqdn_sender,
    reject_unknown_sender_domain,
    check_sender_access hash:/etc/postfix/sender_access

# === RESTRICTIONS RECIPIENT ===
smtpd_recipient_restrictions =
    permit_mynetworks,
    permit_sasl_authenticated,
    reject_unauth_destination,
    reject_non_fqdn_recipient,
    reject_unknown_recipient_domain,
    reject_rbl_client zen.spamhaus.org

# === RATE LIMITING ===
smtpd_client_connection_count_limit = 10
smtpd_client_connection_rate_limit = 30
smtpd_client_message_rate_limit = 100
anvil_rate_time_unit = 60s

# === SÉCURITÉ GÉNÉRALE ===
disable_vrfy_command = yes
smtpd_delay_reject = yes
```

Appliquer la configuration

```
# Vérifier la configuration  
postfix check
```

```
# Recharger Postfix  
sudo systemctl reload postfix
```

Ordre des restrictions

Important : L'ordre compte !

1. permit_mynetworks en premier (vos serveurs passent)
2. permit_sasl_authenticated ensuite (utilisateurs authentifiés passent)
3. reject_unauth_destination avant les RBL (économise des requêtes DNS)
4. RBL en dernier (coûteux en requêtes DNS)



Monitoring de l'efficacité

Statistiques des rejets

```
# Compter les rejets par type  
sudo grep 'reject:' /var/log/mail.log | \  
awk '{print $7}' | sort | uniq -c | sort -rn
```



Exemple de sortie :

```
1523 RCPT  
842 HELO  
312 Client  
156 Sender
```



Top des IPs rejetées

```
sudo grep 'reject:' /var/log/mail.log | \
grep -oP '\[(\d+\.\d+\.\d+\.\d+)\]' | \
sort | uniq -c | sort -rn | head -20
```



🚫 Raisons de rejet

```
sudo grep 'reject:' /var/log/mail.log | tail -20
```

Troubleshooting rapide

✖ Problème : Emails légitimes rejetés

Diagnostic :

```
sudo grep 'reject:' /var/log/mail.log | tail -20
```



Solution :

1. Identifier la règle responsable
2. Whitelister l'IP ou le domaine légitime
3. Recharger Postfix



Exemple de whitelist :

```
# /etc/postfix/sender_access  
partenaire-important.com      OK  
  
# /etc/postfix/client_access  
203.0.113.50                  OK
```



```
sudo postmap /etc/postfix/sender_access  
sudo postmap /etc/postfix/client_access  
sudo systemctl reload postfix
```



Exercices pratiques

🎯 Exercice 1 : Configuration RBL

Objectif : Mettre en place la protection RBL avec Spamhaus

Tâches :

1. Ajouter Spamhaus ZEN dans `smtpd_recipient_restrictions`
2. Recharger Postfix
3. Tester avec une IP blacklistée : `dig 4.3.2.1.zen.spamhaus.org`
4. Consulter les logs pour voir les rejets



Commandes utiles :

```
sudo nano /etc/postfix/main.cf
sudo systemctl reload postfix
sudo grep 'reject_rbl_client' /var/log/mail.log
```



🎯 Exercice 2 : Restrictions HELO complètes

Objectif : Bloquer les HELO invalides et suspects

Tâches :

1. Configurer les restrictions HELO dans main.cf
2. Créer le fichier /etc/postfix/helo_access
3. Ajouter votre domaine pour rejeter les usurpations
4. Tester avec telnet

Test avec telnet :

```
telnet localhost 25
HELO localhost
# Doit être rejeté car non-FQDN

HELO votre-domaine.com
# Doit être rejeté car usurpation
```



Vérifier les logs :

```
sudo grep 'reject.*HELO' /var/log/mail.log
```



Points clés à retenir

Protection anti-spam de base

Couche 1 - Réseau :

- RBL Spamhaus ZEN (le plus fiable)
- Rate limiting (10 connexions, 30/min)

Couche 2 - SMTP :

- Restrictions HELO (rejet HELO invalides)
- Restrictions Sender (domaines valides)
- Restrictions Recipient (éviter open relay)
- Restrictions Client (blacklist/whitelist IPs)

Ordre des règles = important !

1. permit_mynetworks (vos serveurs)
2. permit_sasl_authenticated (utilisateurs authentifiés)
3. reject_unauth_destination (anti open relay)
4. Restrictions diverses
5. reject_rbl_client en dernier (coûteux)

💡 Commandes essentielles

```
# Vérifier la configuration  
postfix check  
  
# Recharger config  
sudo systemctl reload postfix  
  
# Voir les rejets  
sudo grep 'reject:' /var/log/mail.log | tail -20  
  
# Statistiques des rejets  
sudo grep 'reject:' /var/log/mail.log | \  
awk '{print $7}' | sort | uniq -c | sort -rn
```

💡 Équilibre et ajustement

- Trop strict → emails légitimes rejetés
- Trop laxiste → spam qui passe

Solution : Monitorer les logs et ajuster progressivement

Whitelist : Toujours possible pour les partenaires de confiance

Prochaine étape

La protection anti-spam de base est en place ! Maintenant, attaquons les logs !

Module suivant : Logs et monitoring →



QCM - Module 7 : Protection anti-spam

Question 1

Que signifie RBL ?

- A) Real-time Block List
- B) Realtime Blackhole List
- C) Reverse Blacklist
- D) Reject Bad Links

Question 2

Quel est le principe du greylisting ?

- A) Bloquer tous les emails gris
- B) Rejeter temporairement les nouveaux expéditeurs
- C) Mettre en liste grise les spameurs
- D) Filtrer les emails sans couleur

Question 3

Quelle restriction doit TOUJOURS être présente pour éviter un open relay ?

- A) reject_invalid_hostname
- B) reject_non_fqdn_sender
- C) reject_unauth_destination
- D) reject_unknown_sender_domain



Question 4

Quel service Postfix filtre les connexions SMTP entrantes avant de passer la main à `smtpd` pour bloquer les clients suspects ?

- A) anvil
- B) postscreen
- C) cleanup
- D) tlsproxy

Question 5

Quel mot-clé place immédiatement votre réseau local en liste blanche dans `smtpd_recipient_restrictions` ?

- A) `permit_sasl_authenticated`
- B) `permit_mynetworks`
- C) `check_policy_service`
- D) `reject_unknown_client_hostname`

Réponses - Module 7

Question 1 : Réponse B - RBL = Realtime Blackhole List. Ce sont des listes noires DNS d'IPs connues pour envoyer du spam.

Question 2 : Réponse B - Le greylisting **rejette temporairement** (code 450) les emails d'expéditeurs inconnus. Les serveurs légitimes réessaient, les spameurs abandonnent.

Question 3 : Réponse C - `reject_unauth_destination` est **ESSENTIEL** ! Sans elle, votre serveur accepte d'envoyer des emails vers n'importe quel domaine = **OPEN RELAY** !

Question 4 : Réponse B - `postscreen` réalise des tests de réputation et de protocole sur les nouvelles connexions TCP avant d'autoriser l'accès complet à `smtpd`.

Question 5 : Réponse B - `permit_mynetworks` autorise sans délai les clients définis dans `mynetworks`. À placer en premier pour vos administrateurs ou relais internes.



Exercice pratique - Module 7

🎯 Objectif

Mettre en place une protection anti-spam basique

📋 Tâches (25 minutes)

1. Configurer les RBL :

```
sudo postconf -e "smtpd_recipient_restrictions = \
permit_mynetworks, \
permit_sasl_authenticated, \
reject_unauth_destination, \
reject_rbl_client zen.spamhaus.org, \
reject_rbl_client bl.spamcop.net"
sudo systemctl reload postfix
```

2. Ajouter rate limiting :

```
sudo postconf -e "smtpd_client_connection_count_limit = 10"
sudo postconf -e "smtpd_client_connection_rate_limit = 30"
sudo postconf -e "anvil_rate_time_unit = 60s"
sudo systemctl reload postfix
```

3. Tester une IP :

```
dig 4.3.2.1.zen.spamhaus.org  
# Si réponse = IP blacklistée
```

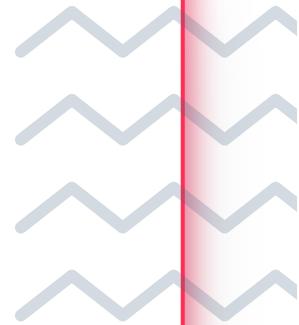
4. Vérifier les logs :

```
sudo tail -f /var/log/mail.log | grep reject
```



Bonus : Installez et configurez Postgrey pour le greylisting





Logs et Surveillance



Analyser, surveiller et comprendre votre serveur mail

Introduction aux logs

Les logs sont votre **meilleur ami** pour :

- Comprendre ce qui se passe
- Diagnostiquer les problèmes
- Détecter les attaques
- Optimiser les performances
- Auditer l'activité



Emplacement des logs

Ubuntu/Debian

```
/var/log/mail.log      # Tous les logs mail  
/var/log/mail.err      # Erreurs uniquement  
/var/log/mail.warn     # Avertissements
```

Rocky Linux / Red Hat

```
/var/log/maillog      # Tous les logs mail
```

Postfix 3.4+

Postfix peut logger directement dans son propre fichier :

```
# Dans main.cf  
maillog_file = /var/log/postfix.log
```

Structure d'une ligne de log

```
Dec 13 10:30:15 mail postfix/smtpd[1234]: ABC123DEF: client=example.com[1.2.3.4]
```

Décryptage : Dec 13 10:30:15 (Date et heure) - mail (Hostname du serveur) - postfix/smtpd[1234] (Service Postfix et PID) - ABC123DEF (Queue ID du message) - client=example.com[1.2.3.4] (Informations supplémentaires)



Suivre un message de bout en bout

Tous les logs d'un même message partagent le même **Queue ID**.



🔍 Exemple : Message ABC123DEF

```
Dec 13 10:30:15 mail postfix/smtpd[1234]: ABC123DEF: client=sender.com[1.2.3.4]
Dec 13 10:30:15 mail postfix/cleanup[1235]: ABC123DEF: message-id=<test@sender.com>
Dec 13 10:30:15 mail postfix/qmgr[1236]: ABC123DEF: from=<user@sender.com>, size=1234
Dec 13 10:30:16 mail postfix/smtp[1237]: ABC123DEF: to=<dest@example.com>, status=sent
Dec 13 10:30:16 mail postfix/qmgr[1236]: ABC123DEF: removed
```

Parcours complet :

- 1. **smtpd** (Message reçu de sender.com)
- 2. **cleanup** (Nettoyage et attribution d'un message-id)
- 3. **qmgr** (Mise en queue, expéditeur et taille)
- 4. **smtp** (Envoi réussi vers dest@example.com)
- 5. **qmgr** (Message supprimé de la queue)

Rechercher un message

```
sudo grep ABC123DEF /var/log/mail.log
```

Types de messages courants

✓ Messages réussis

```
postfix/smtp[1234]: ABC123: to=<user@example.com>, relay=mail.example.com[1.2.3.4]:25, delay=0.52, delays=0.01/0/0.02/0.49, dsn=2.0.0, status=sent  
(250 2.0.0 Ok: queued as DEF456)
```

Informations clés : to (Destinataire) - relay (Serveur qui a accepté l'email) - delay (Délai total en secondes) - delays (Détails a/b/c/d : temps avant queue/dans queue/connexion/transmission) - dsn (Code de statut 2.x.x = succès) - status=sent (Envoi réussi)



Messages en différé

```
postfix/smtp[1234]: ABC123: to=<user@example.com>, relay=mail.example.com[1.2.3.4]:25, delay=5.2, delays=0.01/5/0.02/0.17, dsn=4.4.1, status=deferred
(connect to mail.example.com[1.2.3.4]:25: Connection timed out)
```

Code 4.x.x = Erreur temporaire - Le message sera réessayé plus tard.

Messages rejetés

```
postfix/smtpd[1234]: NOQUEUE: reject: RCPT from unknown[1.2.3.4]: 554 5.7.1 <spammer@spam.com>; Sender address rejected: Access denied; from=<spammer@spam.com> to=<user@example.com>
```

NOQUEUE : Message rejeté avant même d'entrer en queue - **554 5.7.1** : Code d'erreur permanente - **Sender address rejected** : Raison du rejet

Messages rebondis (bounce)

```
postfix/smtp[1234]: ABC123: to=<invalid@nonexistent.com>, relay=mail.nonexistent.com[1.2.3.4]:25, delay=2.5, delays=0.01/0/0.5/2, dsn=5.1.1,
status=bounced (host mail.nonexistent.com[1.2.3.4] said: 550 5.1.1 <invalid@nonexistent.com>; Recipient address rejected: User unknown)
```

Code 5.x.x = Erreur permanente - Un email de bounce sera envoyé à l'expéditeur.

Commandes utiles



Suivre les logs en temps réel

```
# Ubuntu/Debian  
sudo tail -f /var/log/mail.log  
  
# Rocky Linux  
sudo tail -f /var/log/maillog  
  
# Avec grep pour filtrer  
sudo tail -f /var/log/mail.log | grep postfix
```

Rechercher dans les logs

Par Queue ID
`sudo grep ABC123 /var/log/mail.log`

Par adresse email
`sudo grep "user@example.com" /var/log/mail.log`

Erreurs uniquement
`sudo grep "error\|warning\|fatal" /var/log/mail.log`



Statistiques

```
# Compter les messages envoyés aujourd'hui  
sudo grep "$(date +%b %e)" /var/log/mail.log | grep "status=sent" | wc -l  
  
# Compter les rejets  
sudo grep "reject:" /var/log/mail.log | wc -l
```

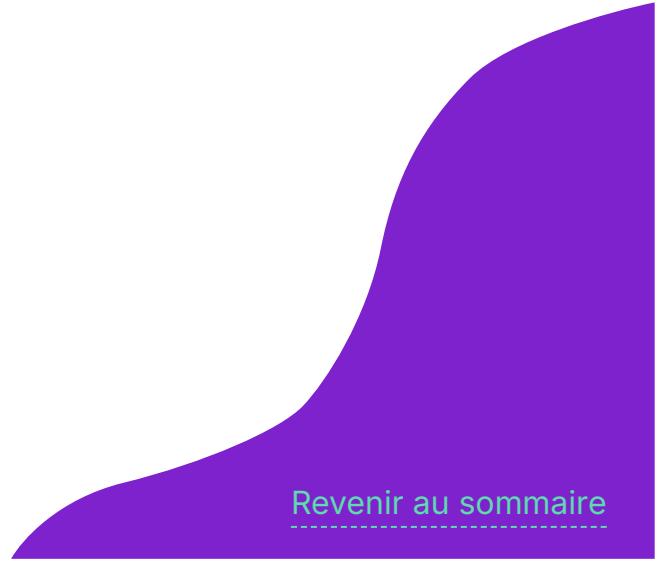
[Revenir au sommaire](#)





Messages par heure

```
sudo grep "status=sent" /var/log/mail.log | awk '{print $1, $2, $3}' | uniq -c
```



[Revenir au sommaire](#)

TOP Top des domaines destinataires

```
sudo grep "status=sent" /var/log/mail.log | grep -oP 'to=<[^>]+>' | sed 's/.*/@/' | sed 's/>//' | sort | uniq -c | sort -rn | head -20
```

🚫 Top des rejets

```
sudo grep "reject:" /var/log/mail.log | awk '{print $7}' | sort | uniq -c | sort -rn
```



Outils d'analyse de logs

pflogsumm

Générateur de rapports pour Postfix.



Installation

```
# Ubuntu/Debian  
sudo apt install pflogsumm  
  
# Rocky Linux  
sudo dnf install postfix-perl-scripts
```



Générer un rapport

```
# Rapport du jour  
sudo pflogsumm /var/log/mail.log  
  
# Rapport de la veille  
sudo pflogsumm /var/log/mail.log.1  
  
# Rapport détaillé  
sudo pflogsumm -d today /var/log/mail.log
```



Exemple de sortie :

Grand Totals

```
-----  
messages received:      1234  
messages sent:         1156  
messages deferred:     45  
messages bounced:      12  
messages rejected:     789  
bytes received:        5.2 MB  
bytes sent:            4.8 MB
```



⌚ Automatiser avec cron

```
# Rapport quotidien envoyé par email
0 6 * * * /usr/sbin/pflogsumm -d yesterday /var/log/mail.log | mail -s "Postfix Report" admin@example.com
```



Logwatch

Analyseur de logs système général (pas que Postfix).



Installation

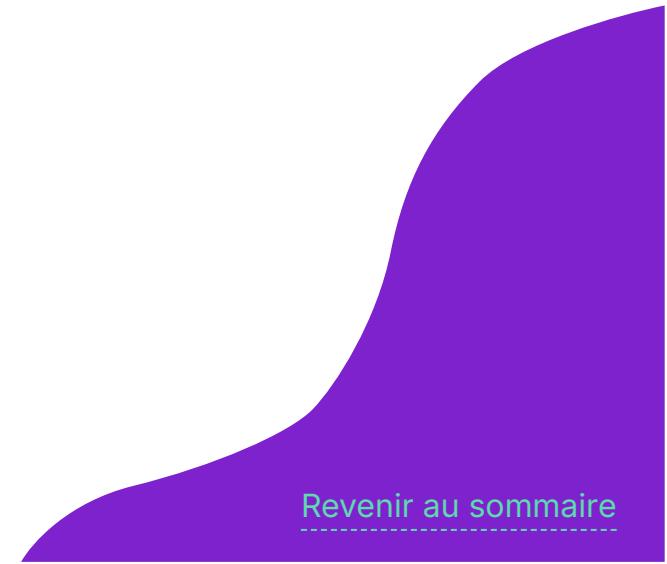
```
# Ubuntu/Debian  
sudo apt install logwatch
```

```
# Rocky Linux  
sudo dnf install logwatch
```



Rapport Postfix

```
sudo logwatch --service postfix --range today --detail high
```



Revenir au sommaire



GoAccess

Analyseur de logs en temps réel avec interface web.

(Plus orienté Apache/Nginx, mais peut s'adapter)



Elasticsearch + Kibana

Pour les gros volumes, centralisez les logs dans Elasticsearch et visualisez avec Kibana.

(Avancé, pour le module perfectionnement)



Surveillance en temps réel

Monitoring basique avec scripts

Script de surveillance

```
#!/bin/bash
# monitor-postfix.sh

echo "==== État de Postfix ==="
systemctl status postfix --no-pager | head -3
echo ""

echo "==== Queue ==="
mailq | tail -1
echo ""

echo "==== Activité récente (5 dernières minutes) ==="
```

```
SINCE=$(date -d '5 minutes ago' '+%b %e %H:%M')
echo "Envoyés : $(sudo grep "$SINCE" /var/log/mail.log | grep "status=sent" | wc -l)"
echo "Rejetés : $(sudo grep "$SINCE" /var/log/mail.log | grep "reject:" | wc -l)"
echo "Différés : $(sudo grep "$SINCE" /var/log/mail.log | grep "status=deferred" | wc -l)"
```

⌚ Automatiser

Toutes les 5 minutes

```
*/5 * * * * /usr/local/bin/monitor-postfix.sh
```



Alertes automatiques

💡 Alerter si queue trop grosse

```
#!/bin/bash
# alert-queue.sh

QUEUE_SIZE=$(mailq | tail -1 | awk '{print $5}')
THRESHOLD=1000

if [ "$QUEUE_SIZE" -gt "$THRESHOLD" ]; then
    echo "Queue size: $QUEUE_SIZE" | mail -s "ALERT: Postfix queue" admin@example.com
fi
```



⚠️ Alerter si trop de rejets

```
#!/bin/bash
# alert-rejects.sh

REJECTS=$(sudo grep "$(date +\%b\ %e)" /var/log/mail.log | grep "reject:" | wc -l)
THRESHOLD=500

if [ "$REJECTS" -gt "$THRESHOLD" ]; then
    echo "Rejects today: $REJECTS" | mail -s "ALERT: High rejects" admin@example.com
fi
```



Monitoring avec Prometheus + Grafana

📦 Installation de l'exporteur Postfix

```
git clone https://github.com/kumina/postfix_exporter  
cd postfix_exporter  
go build  
sudo cp postfix_exporter /usr/local/bin/
```



Créer un service systemd

```
[Unit]
Description=Postfix Exporter
After=network.target

[Service]
Type=simple
User=postfix
ExecStart=/usr/local/bin/postfix_exporter \
--postfix.logfile_path=/var/log/mail.log
Restart=always

[Install]
WantedBy=multi-user.target
```

```
sudo systemctl enable postfix-exporter  
sudo systemctl start postfix-exporter
```



Configuration Prometheus

```
scrape_configs:  
  - job_name: 'postfix'  
    static_configs:  
      - targets: ['localhost:9154']
```



Dashboard Grafana

Importer le dashboard ID **11733** depuis Grafana.com

Vous aurez de magnifiques graphiques ! 📈



Rotation des logs

Les logs peuvent devenir très gros !

Il faut les **rotationner** (archiver et compresser).



Configuration logrotate

Fichier /etc/logrotate.d/postfix :

```
/var/log/mail.log {  
    daily  
    rotate 30  
    missingok  
    notifempty  
    compress  
    delaycompress  
    sharedscripts  
    postrotate  
        systemctl reload postfix > /dev/null  
    endscript  
}
```



Explication :

- daily : Rotation quotidienne
- rotate 30 : Garder 30 jours d'archives
- missingok : Pas d'erreur si fichier manquant
- notifempty : Ne pas rotationner si vide
- compress : Compresser les archives
- delaycompress : Ne pas compresser immédiatement
- postrotate : Commande après rotation



Tester la rotation

```
sudo logrotate -f /etc/logrotate.d/postfix
```



Debugging avancé

Activer le mode verbose

```
# Dans main.cf
debug_peer_list = example.com, 1.2.3.4
debug_peer_level = 2
```



```
sudo systemctl reload postfix
```

Tous les échanges avec example.com ou 1.2.3.4 seront loggés en détail.



Tracer un processus spécifique

```
# Suivre les logs d'un processus  
sudo tail -f /var/log/mail.log | grep "smtpd\[1234\]"
```



Inspecter un message dans la queue

```
# Voir le contenu complet  
sudo postcat -q ABC123DEF
```

```
# Voir seulement les headers  
sudo postcat -q ABC123DEF | head -50
```



Exercices pratiques

Exercice 1 : Suivre un message

1. Envoyez un email de test
2. Notez son Queue ID dans les logs
3. Tracez son parcours complet avec grep

🎯 Exercice 2 : Statistiques

1. Comptez les emails envoyés aujourd'hui
2. Comptez les emails rejetés
3. Identifiez les 5 domaines destinataires les plus fréquents

Exercice 3 : pflogsumm

1. Installez pflogsumm
2. Générez un rapport du jour
3. Analysez les résultats

🎯 Exercice 4 : Script de monitoring

1. Créez un script qui affiche :

- État de Postfix
- Taille de la queue
- Nombre d'emails envoyés dans la dernière heure

2. Automatisez-le avec cron

🎯 Exercice 5 : Alertes

1. Créez un script qui alerte si la queue dépasse 100 messages
2. Testez-le en créant des messages en deferred
3. Vérifiez la réception de l'alerte

Points clés à retenir

💡 Logs

Emplacement :

- Ubuntu/Debian : /var/log/mail.log
- Rocky Linux : /var/log/maillog

Queue ID : Identifiant unique pour tracer un message

💡 Commandes essentielles

```
# Suivre en temps réel  
sudo tail -f /var/log/mail.log
```

```
# Rechercher  
sudo grep QUEUE_ID /var/log/mail.log
```

```
# Statistiques  
pflogsumm /var/log/mail.log
```



Codes de statut

2.x.x : Succès

4.x.x : Erreur temporaire (deferred)

5.x.x : Erreur permanente (bounce)

Surveillance

Automatiser : Scripts + cron

Alerter : Mail si problème

Analyser : pflogsumm, Logwatch

Visualiser : Prometheus + Grafana

Prochaine étape

Vous savez maintenant surveiller votre serveur ! Passons à la **sauvegarde et restauration** pour sécuriser vos données ! 

Module suivant : Sauvegarde et restauration →



QCM - Module 10 : Logs et surveillance

Question 1

Où se trouvent les logs Postfix sur Ubuntu/Debian ?

- A) /var/log/postfix.log
- B) /var/log/mail.log
- C) /var/log/messages
- D) /var/log/syslog

Question 2

Que signifie le code dsn=2.0.0 dans les logs ?

- A) Erreur permanente
- B) Erreur temporaire
- C) Succès
- D) Message en attente

Question 3

Quelle commande affiche en temps réel les rejets de spam ?

- A) mailq
- B) tail -f /var/log/mail.log | grep reject
- C) postqueue -p
- D) postsuper -v

Question 4

Quelle commande basée sur `journalctl` affiche en direct les logs Postfix sur une distribution systemd ?

- A) `journalctl postfix -f`
- B) `journalctl -u postfix -f`
- C) `journalctl --follow mail.log`
- D) `journalctl -t postfix/smtpd`

Question 5

Quel est le nom de l'identifiant unique qui permet de suivre un email dans les logs et les files d'attente ?

- A) PID
- B) Queue ID
- C) UID
- D) RID

Réponses - Module 10

Question 1 : Réponse B - Sur Ubuntu/Debian : `/var/log/mail.log` . Sur Rocky/RHEL : `/var/log/maillog` .

Question 2 : Réponse C - `dsn=2.x.x` = **Succès** ! `dsn=4.x.x` = Erreur temporaire, `dsn=5.x.x` = Erreur permanente.

Question 3 : Réponse B - `tail -f /var/log/mail.log | grep reject` affiche en **temps réel** tous les messages rejetés.

Question 4 : Réponse B - `journalctl -u postfix -f` suit en temps réel les journaux du service `systemd postfix` , pratique quand `/var/log/mail.log` est géré par `rsyslog` ou `systemd-journald` .

Question 5 : Réponse B - Le **Queue ID** (ex : `3F2A641234`) est attribué par Postfix dès la réception et reste visible dans les logs, `mailq` , `postcat` , etc.

Exercice pratique - Module 10

🎯 Objectif

Analyser les logs Postfix

📋 Tâches (20 minutes)

1. Suivre les logs en temps réel :

```
sudo tail -f /var/log/mail.log  
# Dans un autre terminal, envoyez un email de test
```

2. Rechercher un message spécifique :

```
# Récupérer le Queue ID depuis mailq  
mailq  
# Rechercher dans les logs  
sudo grep "VOTRE_QUEUE_ID" /var/log/mail.log
```

3. Analyser les erreurs :

```
sudo grep "status=deferred" /var/log/mail.log | tail -20  
sudo grep "status=bounced" /var/log/mail.log | tail -20
```

4. Statistiques des rejets :

```
sudo grep "reject:" /var/log/mail.log | wc -l  
sudo grep "reject_rbl_client" /var/log/mail.log | wc -l
```

5. Utiliser pflogsumm (si installé) :

```
sudo apt install pflogsumm # Ubuntu/Debian  
sudo pflogsumm /var/log/mail.log | less
```

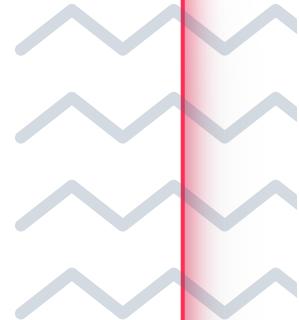
Bonus : Créez un script pour générer un rapport quotidien des logs



Sauvegarde et Restauration



Protéger vos données et préparer le disaster recovery



Introduction

Un serveur mail contient des données **critiques** :

- Emails des utilisateurs
- Configuration Postfix
- Clés DKIM
- Certificats SSL
- Bases de données (si domaines virtuels)

Sans sauvegarde = Catastrophe assurée !

Analogie

Imaginez perdre tous vos emails professionnels des 5 dernières années en une seconde... **Les sauvegardes ne sont pas optionnelles !**



Quoi sauvegarder ?

Les 4 éléments critiques

- 1. Configuration Postfix :** /etc/postfix/ (main.cf, master.cf, tables...)
- 2. Clés DKIM :** /etc/opendkim/
- 3. Certificats SSL :** /etc/letsencrypt/
- 4. Mailboxes :** /var/mail/vhosts/

Optionnel

Queue Postfix : /var/spool/postfix/ (change constamment, rarement sauvegardé)

Bases de données : Si vous utilisez MySQL/PostgreSQL pour domaines virtuels

Stratégie de sauvegarde

Règle 3-2-1

3 copies - **2** supports différents - **1** copie hors site



Fréquence

- Configuration : **Quotidienne**
- Mailboxes : **2-4 fois/jour** (selon criticité)
- Bases de données : **Quotidienne**



Rétention simple

- **7 jours** : Sauvegardes quotidiennes
- **4 semaines** : Sauvegardes hebdomadaires
- **1 an** : Sauvegardes mensuelles



Méthodes de sauvegarde

Méthode 1 : tar (simple)

📦 Créer une sauvegarde

```
#!/bin/bash
DATE=$(date +%Y%m%d-%H%M%S)
BACKUP_DIR="/backup/postfix"
mkdir -p $BACKUP_DIR

# Tout en un
sudo tar czf $BACKUP_DIR/postfix-full-$DATE.tar.gz \
/etc/postfix \
/etc/opendkim \
/etc/letsencrypt \
/var/mail/vhosts

echo "✅ Backup completed: $DATE"
```

Restaurer

```
sudo tar xzf postfix-full-XXXXXX.tar.gz -C /
sudo systemctl reload postfix
```



Méthode 2 : rsync (incrémental)

Avantage : Seuls les fichiers modifiés sont copiés (rapide !)

```
#!/bin/bash
BACKUP_DIR="/backup/postfix"
DATE=$(date +%Y%m%d)

# Sauvegarde locale
rsync -avz --delete /etc/postfix/ $BACKUP_DIR/$DATE/postfix/
rsync -avz --delete /etc/opendkim/ $BACKUP_DIR/$DATE/opendkim/
rsync -avz --delete /var/mail/vhosts/ $BACKUP_DIR/$DATE/mailboxes/
```

Sauvegarde distante

```
# Vers un autre serveur (nécessite clé SSH)
rsync -avz -e ssh /etc/postfix/ backup@server.com:/backups/postfix/
```



Sauvegarde des bases de données (optionnel)

Si vous utilisez MySQL/PostgreSQL pour les domaines virtuels :

```
# MySQL  
mysqldump -u root -p mailserver | gzip > /backup/maildb-$(date +%Y%m%d).sql.gz  
  
# PostgreSQL  
sudo -u postgres pg_dump mailserver | gzip > /backup/maildb-$(date +%Y%m%d).sql.gz
```



Automatisation avec cron

Planifier les sauvegardes

```
sudo crontab -e
```



```
# Sauvegarde quotidienne à 2h du matin  
0 2 * * * /usr/local/bin/backup-postfix.sh  
  
# Sauvegarde toutes les 6 heures des mailboxes  
0 */6 * * * /usr/local/bin/backup-mailboxes.sh  
  
# Sauvegarde hebdomadaire complète le dimanche à 3h  
0 3 * * 0 /usr/local/bin/backup-complete.sh
```



Restauration

Procédure simple

```
#!/bin/bash
# restore-postfix.sh

# 1. Arrêter les services
sudo systemctl stop postfix opendkim

# 2. Restaurer depuis l'archive
sudo tar xzf /backup/postfix/postfix-full-*.tar.gz -C /

# 3. Corriger les permissions
sudo chown -R postfix:postfix /etc/postfix
sudo chown -R opendkim:opendkim /etc/opendkim
sudo chown -R vmail:vmail /var/mail/vhosts

# 4. Recomplier les tables
sudo postmap /etc/postfix/virtual
sudo newaliases

# 5. Vérifier et redémarrer
sudo postfix check
sudo systemctl start opendkim postfix

echo "✅ Restauration terminée !"
```



Tester les sauvegardes !

⚠ Une sauvegarde non testée = Une sauvegarde qui ne marche pas !

Plan : Testez la restauration tous les 3 mois sur une VM



Pour aller plus loin

Les outils avancés suivants seront vus dans la formation **Perfectionnement** :

- **Borg Backup** : Sauvegarde dédupliquée et chiffrée
- **Rclone vers Cloud** : S3, Google Cloud, Backblaze
- **Bacula** : Solution d'entreprise complète
- **Restic** : Alternative moderne à Borg

Monitoring simple

Vérifier les sauvegardes

```
# Lister les sauvegardes récentes  
ls -lh /backup/postfix/ | tail -10
```

[Revenir au sommaire](#)



```
# Vérifier la taille  
du -sh /backup/postfix/
```



Alerte simple avec cron

```
# Dans le script de sauvegarde, ajouter :  
if [ $? -eq 0 ]; then  
    echo "✓ Backup OK" | mail -s "Backup Success" admin@example.com  
else  
    echo "✗ Backup FAILED!" | mail -s "ALERT: Backup FAILED" admin@example.com  
fi
```

Checklist de sauvegarde

- Configuration Postfix (/etc/postfix/)
- Clés DKIM (/etc/opendkim/)
- Certificats SSL (/etc/letsencrypt/)
- Mailboxes (/var/mail/vhosts/)
- Bases de données (si utilisées)



- Sauvegardes automatisées (cron)
- Sauvegardes hors site (cloud/serveur distant)
- Tests de restauration réguliers
- Monitoring et alertes
- Documentation de la procédure

Exercices pratiques

🎯 Exercice 1 : Sauvegarde et restauration

Objectif : Créer une sauvegarde et la restaurer

Tâches :

1. Créez un script de sauvegarde avec tar pour /etc/postfix/
2. Exécutez le script et vérifiez l'archive
3. Modifiez un fichier de config (ex: main.cf)
4. Restaurez depuis la sauvegarde
5. Vérifiez que la modification a été annulée



Commandes utiles :

```
# Créer la sauvegarde
sudo tar czf /backup/postfix-$(date +%Y%m%d).tar.gz /etc/postfix/

# Vérifier l'archive
tar tzf /backup/postfix-*.tar.gz | head

# Restaurer
sudo tar xzf /backup/postfix-*.tar.gz -C /
```



🎯 Exercice 2 : Automatisation

Objectif : Automatiser les sauvegardes quotidiennes

Tâches :

1. Créez le script /usr/local/bin/backup-postfix.sh
2. Rendez-le exécutable
3. Configurez un cron pour l'exécuter à 2h du matin
4. Testez manuellement le script
5. Vérifiez les logs cron le lendemain



Configuration cron :

```
sudo crontab -e  
# Ajouter :  
0 2 * * * /usr/local/bin/backup-postfix.sh  
  
# Vérifier les logs  
sudo grep CRON /var/log/syslog | grep backup
```



Points clés à retenir

💡 Essentiel

Quoi sauvegarder :

- Configuration : /etc/postfix/ , /etc/opendkim/
- Certificats : /etc/letsencrypt/
- Mailboxes : /var/mail/vhosts/
- Bases de données (si utilisées)

Outils de base :

- `tar` : Simple et universel
- `rsync` : Incrémental et rapide

Règle 3-2-1 :

- 3 copies, 2 supports différents, 1 copie hors site

Automatisation :

- Cron pour planifier
- Alertes en cas d'échec
- **Tester les restaurations !**



Prochaine étape

Vos données sont maintenant protégées ! Passons aux **exercices pratiques** pour consolider toutes ces connaissances ! 

Module suivant : Exercices pratiques débutant →



QCM - Module 11 : Sauvegarde et restauration

Question 1

Quels sont les 3 éléments ESSENTIELS à sauvegarder pour Postfix ?

- A) Logs, queue, processus
- B) Configuration, certificats SSL, DKIM keys
- C) Binaires, man pages, documentation
- D) Cache DNS, cookies, sessions

Question 2

Quelle est la règle 3-2-1 de sauvegarde ?

- A) 3 serveurs, 2 datacenters, 1 cloud
- B) 3 copies, 2 supports différents, 1 copie hors site
- C) 3 jours, 2 semaines, 1 mois
- D) 3 fichiers, 2 formats, 1 compression

Question 3

Faut-il sauvegarder la queue Postfix ?

- A) Oui, toujours
- B) Non, elle change constamment
- C) Seulement la queue active
- D) Seulement en production

Question 4

Quelle commande permet de sauvegarder la configuration active de Postfix dans un fichier texte versionné ?

- A) postfix status > postfix.conf
- B) postconf -n > postfix.conf
- C) postqueue -p > postfix.conf
- D) systemctl show postfix > postfix.conf

Question 5

Quel service du système planifie l'exécution de votre script de sauvegarde chaque nuit à 02h00 ?

- A) systemd-networkd
- B) cron
- C) logrotate
- D) cupsd



Réponses - Module 11

Question 1 : Réponse B - Les 3 essentiels : Configuration (`/etc/postfix/`), certificats SSL (`/etc/letsencrypt/`), clés DKIM (`/etc/opendkim/keys/`).

Question 2 : Réponse B - Règle 3-2-1 : 3 copies de vos données, sur 2 supports différents, avec 1 copie hors site.

Question 3 : Réponse B - La queue change **constamment** et se vide naturellement. Sauvegarder la config est plus important que la queue.

Question 4 : Réponse B - `postconf -n` liste uniquement les paramètres personnalisés. Redirigez la sortie vers un fichier pour tracer vos changements dans Git ou une sauvegarde.

Question 5 : Réponse B - `cron` (ou `cronie`) reste la méthode la plus simple pour lancer un script récurrent. Vous pouvez aussi créer un timer `systemd` si vous préférez.



Exercice pratique - Module 11

🎯 Objectif

Sauvegarder et restaurer la configuration Postfix

📋 Tâches (15 minutes)

1. Créer un script de sauvegarde :

```
#!/bin/bash
# backup-postfix.sh
DATE=$(date +\%Y\%m\%d-\%H\%M\%S)
BACKUP_DIR="/backup/postfix"
mkdir -p $BACKUP_DIR
# Sauvegarder la configuration
tar -czf $BACKUP_DIR/postfix-config-$DATE.tar.gz /etc/postfix/
```

```
# Sauvegarder les certificats SSL
tar -czf $BACKUP_DIR/ssl-certs-$DATE.tar.gz /etc/letsencrypt/

# Sauvegarder les clés DKIM (si présentes)
if [ -d /etc/opendkim/keys ]; then
    tar -czf $BACKUP_DIR/dkim-keys-$DATE.tar.gz /etc/opendkim/keys/
fi

echo "Backup completed: $DATE"
ls -lh $BACKUP_DIR/
```



2. Rendre exécutable et tester :

```
chmod +x backup-postfix.sh  
sudo ./backup-postfix.sh
```



3. Simuler une restauration :

```
# Copier la config actuelle  
sudo cp -r /etc/postfix /etc/postfix.backup  
  
# Restaurer depuis l'archive  
cd /  
sudo tar -xzf /backup/postfix/postfix-config-VOTRE_DATE.tar.gz  
  
# Vérifier  
postfix check  
sudo systemctl reload postfix
```

4. Automatiser avec cron :

```
# Ajouter dans crontab  
sudo crontab -e  
# Ajouter : 0 2 * * * /chemin/vers/backup-postfix.sh
```

Bonus : Envoyez les sauvegardes vers un serveur distant avec rsync ou scp



Exercices Pratiques Débutant

🎯 Mettre en pratique les connaissances acquises

[Revenir au sommaire](#)

Introduction

Ces exercices couvrent tous les modules de la formation **initiation**.

Chaque exercice est conçu pour renforcer un concept spécifique.

Organisation

Les exercices sont classés par **difficulté** :

- **★ Facile** : Configuration de base
- **★★ Moyen** : Configuration avancée
- **★★★ Difficile** : Troubleshooting et intégration

Exercice 1 : Installation de base ★

Objectif

Installer et configurer un serveur Postfix fonctionnel.

Prérequis

- Serveur Linux (Ubuntu 22.04+ ou Rocky Linux 9+)
- Accès root/sudo
- Nom de domaine configuré en DNS

Tâches

1. Installez Postfix
2. Configurez `myhostname` , `mydomain` , `myorigin`
3. Configurez `mydestination` pour accepter les emails locaux
4. Testez l'envoi d'un email local
5. Vérifiez que l'email est bien arrivé

✓ Critères de validation

- Postfix démarre sans erreur
- postconf -n affiche votre configuration
- Email local reçu dans /home/user/Maildir/

Solution

```
# 1. Installation  
sudo apt update && sudo apt install postfix -y  
  
# 2. Configuration  
sudo postconf -e "myhostname = mail.example.com"  
sudo postconf -e "mydomain = example.com"  
sudo postconf -e "myorigin = \$mydomain"
```

```
# 3. Destinations  
sudo postconf -e "mydestination = \$myhostname, localhost.\$mydomain, localhost, \$mydomain"  
  
# 4. Maildir  
sudo postconf -e "home_mailbox = Maildir/"  
  
# 5. Recharger  
sudo systemctl reload postfix
```

```
# 6. Test  
echo "Test Postfix" | mail -s "Test" $USER  
  
# 7. Vérifier  
ls ~/Maildir/new/  
cat ~/Maildir/new/*
```



Exercice 2 : Configuration DNS

Objectif

Configurer correctement les enregistrements DNS pour votre serveur mail.

Prérequis

- Accès à la configuration DNS de votre domaine - Serveur Postfix installé

Tâches

1. Créez un enregistrement A pour mail.example.com
2. Créez un enregistrement MX pour example.com
3. Configurez le PTR (reverse DNS)
4. Vérifiez tous les enregistrements

Critères de validation

- dig mail.example.com A retourne votre IP - dig example.com MX retourne mail.example.com - dig -x VOTRE_IP retourne mail.example.com

Solution

```
# Enregistrements DNS à ajouter  
  
# Enregistrement A  
# Type: A  
# Nom: mail.example.com  
# Valeur: 203.0.113.10  
  
# Enregistrement MX  
# Type: MX  
# Nom: example.com  
# Priorité: 10  
# Valeur: mail.example.com
```

```
# PTR (à configurer chez votre hébergeur)
# Type: PTR
# IP: 203.0.113.10
# Valeur: mail.example.com

# Vérification
dig mail.example.com A +short
dig example.com MX +short
dig -x 203.0.113.10 +short
```

Exercice 3 : Alias et redirections

Objectif

Créer des alias pour rediriger les emails.



Tâches

1. Créez un alias contact qui redirige vers votre email
2. Créez un alias support qui redirige vers plusieurs adresses
3. Redirigez tous les emails système (root, postmaster) vers votre email
4. Testez chaque alias

✓ Critères de validation

- Email à contact@localhost arrive à votre adresse
- Email à support@localhost arrive à toutes les adresses
- Email à root@localhost arrive à votre adresse



Solution

```
# Éditer /etc/aliases  
sudo nano /etc/aliases
```



```
# Contenu de /etc/aliases  
  
# Alias simple  
contact: john@example.com  
  
# Alias multiples  
support: john@example.com, jane@example.com, admin@example.com  
  
# Redirections système  
postmaster: admin@example.com  
webmaster: admin@example.com  
root: admin@example.com
```

```
# Recompileur
sudo newaliases

# Tester
echo "Test contact" | mail -s "Test" contact
echo "Test support" | mail -s "Test" support
echo "Test root" | mail -s "Test" root
```



Exercice 4 : Domaines virtuels ★★

Objectif

Configurer des domaines virtuels avec redirections.



Tâches

1. Créez un domaine virtuel virtual.local
2. Configurez admin@virtual.local → votre email
3. Configurez info@virtual.local → votre email
4. Créez un catch-all pour virtual.local
5. Testez les redirections

✓ Critères de validation

- Email à admin@virtual.local arrive
- Email à info@virtual.local arrive
- Email à n'importe quoi@virtual.local arrive (catch-all)



Solution

```
# Créer /etc/postfix/virtual  
sudo nano /etc/postfix/virtual
```



```
# Contenu de /etc/postfix/virtual
# Règles spécifiques
admin@virtual.local      john@example.com
info@virtual.local        john@example.com
# Catch-all (doit être après les règles spécifiques)
@virtual.local            catchall@example.com
```



```
# Configuration main.cf
sudo postconf -e "virtual_alias_domains = virtual.local"
sudo postconf -e "virtual_alias_maps = hash:/etc/postfix/virtual"

# Compiler
sudo postmap /etc/postfix/virtual

# Recharger
sudo systemctl reload postfix
```



```
# Tester
echo "Test admin" | mail -s "Test" admin@virtual.local
echo "Test info" | mail -s "Test" info@virtual.local
echo "Test catch-all" | mail -s "Test" random@virtual.local
```

Exercice 5 : Protection anti-spam basique ★★

Objectif

Mettre en place des protections anti-spam de base.



Tâches

1. Configurez les restrictions HELO
2. Configurez les restrictions sender
3. Configurez les restrictions recipient
4. Ajoutez 2 RBL (Spamhaus et Barracuda)
5. Testez les rejets

Critères de validation

- Email avec HELO invalide est rejeté
- Email depuis domaine inexistant est rejeté
- IP blacklistée est rejetée

Solution

```
# Dans /etc/postfix/main.cf

# Restrictions HELO
smtpd_helo_required = yes
smtpd_helo_restrictions =
    permit_mynetworks,
    reject_invalid_helo_hostname,
    reject_non_fqdn_helo_hostname
```

```
# Restrictions sender
smtpd_sender_restrictions =
    permit_mynetworks,
    reject_non_fqdn_sender,
    reject_unknown_sender_domain

# Restrictions recipient
smtpd_recipient_restrictions =
    permit_mynetworks,
    reject_unauth_destination,
    reject_non_fqdn_recipient,
    reject_unknown_recipient_domain,
    reject_rbl_client zen.spamhaus.org,
    reject_rbl_client b.barracudacentral.org
```

```
# Recharger
sudo postfix check
sudo systemctl reload postfix

# Tester le rejet (depuis un autre serveur)
telnet mail.example.com 25
EHLO invalid
MAIL FROM:<user@nonexistent-domain-12345.com>
```



Exercice 6 : SPF ⭐

🎯 Objectif

Créer et publier un enregistrement SPF.



Tâches

1. Listez tous vos serveurs d'envoi d'emails
2. Créez un enregistrement SPF approprié
3. Publiez-le en DNS
4. Testez avec un outil en ligne

✓ Critères de validation

- dig example.com TXT retourne l'enregistrement SPF
- Test sur <https://mxtoolbox.com/spf.aspx> passe

Solution

```
# Enregistrement DNS à ajouter
# Type: TXT
# Nom: example.com (ou @)
# Valeur: v=spf1 a mx ip4:203.0.113.10 -all

# Vérification
dig example.com TXT +short | grep spf

# Test en ligne
# https://www.kitterman.com/spf/validate.html
```



Exercice 7 : DKIM ★★

Objectif

Configurer la signature DKIM pour vos emails.



Tâches

1. Installez OpenDKIM
2. Générez une paire de clés
3. Configurez Postfix pour signer
4. Publiez la clé publique en DNS
5. Testez la signature

Critères de validation

- OpenDKIM démarre sans erreur
- Clé publique publiée en DNS
- Email de test contient DKIM-Signature:
- Vérification DKIM passe

Solution

```
# 1. Installation  
sudo apt install opendkim opendkim-tools -y  
  
# 2. Générer clés  
sudo mkdir -p /etc/opendkim/keys/example.com  
sudo opendkim-genkey -b 2048 -d example.com -D /etc/opendkim/keys/example.com -s mail -v
```



```
# 3. Permissions  
sudo chown opendkim:opendkim /etc/opendkim/keys/example.com/mail.private  
sudo chmod 600 /etc/opendkim/keys/example.com/mail.private  
  
# 4. Configuration OpenDKIM  
sudo nano /etc/opendkim.conf
```

```
# /etc/opendkim.conf
Mode sv
Domain example.com
Selector mail
KeyFile /etc/opendkim/keys/example.com/mail.private
Socket inet:8891@localhost
Canonicalization relaxed/simple
```



```
# 5. Configuration Postfix
sudo postconf -e "smtpd_milters = inet:localhost:8891"
sudo postconf -e "non_smtpd_milters = inet:localhost:8891"
sudo postconf -e "milter_default_action = accept"

# 6. Démarrer
sudo systemctl enable opendkim
sudo systemctl start opendkim
sudo systemctl reload postfix
```

```
# 7. Publier clé DNS
sudo cat /etc/opendkim/keys/example.com/mail.txt

# Type: TXT
# Nom: mail._domainkey.example.com
# Valeur: v=DKIM1; h=sha256; k=rsa; p=MIIBIjAN...

# 8. Tester
echo "Test DKIM" | mail -s "Test" test@gmail.com
# Vérifier la signature dans les headers
```

Exercice 8 : TLS

Objectif

Activer TLS pour sécuriser les communications.



Tâches

1. Obtenez un certificat Let's Encrypt
2. Configurez TLS dans Postfix
3. Restreignez aux versions TLS 1.2+
4. Testez la connexion TLS

Critères de validation

- Certificat Let's Encrypt valide
- TLS activé sur port 25
- TLS 1.2+ uniquement
- Test openssl s_client réussit



Solution

```
# 1. Installation Certbot  
sudo apt install certbot -y  
  
# 2. Obtenir certificat  
sudo certbot certonly --standalone -d mail.example.com  
  
# 3. Vérifier  
sudo ls -la /etc/letsencrypt/live/mail.example.com/
```

```
# 4. Configuration Postfix
sudo postconf -e "smtpd_tls_cert_file = /etc/letsencrypt/live/mail.example.com/fullchain.pem"
sudo postconf -e "smtpd_tls_key_file = /etc/letsencrypt/live/mail.example.com/privkey.pem"
sudo postconf -e "smtpd_tls_security_level = may"
sudo postconf -e "smtpd_tls_protocols = >=TLSv1.2"
sudo postconf -e "smtp_tls_security_level = may"
sudo postconf -e "smtp_tls_protocols = >=TLSv1.2"
```

```
# 5. Recharger  
sudo systemctl reload postfix  
  
# 6. Tester  
openssl s_client -connect mail.example.com:25 -starttls smtp  
# Vérifier : Protocol : TLSv1.2 ou TLSv1.3
```



Exercice 9 : Analyse des logs

Objectif

Analyser les logs pour comprendre l'activité du serveur.



Tâches

1. Envoyez 5 emails de test
2. Trouvez les Queue IDs dans les logs
3. Tracez le parcours complet d'un message
4. Comptez les emails envoyés aujourd'hui
5. Identifiez les 3 destinataires les plus fréquents

Critères de validation

- Vous pouvez tracer un message de A à Z
- Vous savez compter les emails
- Vous savez extraire des statistiques

Solution

```
# 1. Envoyer des tests
for i in {1..5}; do
    echo "Test $i" | mail -s "Test $i" test@example.com
done

# 2. Trouver les Queue IDs
sudo grep "from=<${whoami}@" /var/log/mail.log | tail -5
```

```
# 3. Tracer un message (remplacez QUEUEID)
sudo grep QUEUEID /var/log/mail.log

# 4. Compter emails aujourd'hui
sudo grep "$(date +%b\ %e)" /var/log/mail.log | grep "status=sent" | wc -l

# 5. Top destinataires
sudo grep "status=sent" /var/log/mail.log | \
grep -oP 'to=<[^>]+>' | \
sed 's/.*/@/' | sed 's/>//' | \
sort | uniq -c | sort -rn | head -3
```

Exercice 10 : Sauvegarde ★★

Objectif

Créer un système de sauvegarde automatique.



Tâches

1. Créez un script de sauvegarde
2. Sauvegardez /etc/postfix/ et /etc/opendkim/
3. Testez manuellement le script
4. Automatisez avec cron (quotidien à 2h)
5. Testez la restauration

Critères de validation

- Script crée une archive avec date
- Cron exécute le script quotidiennement
- Restauration fonctionne

Solution

```
#!/bin/bash
# /usr/local/bin/backup-postfix.sh

DATE=$(date +%Y%m%d-%H%M%S)
BACKUP_DIR="/backup/postfix"
mkdir -p $BACKUP_DIR

# Configuration Postfix
tar czf $BACKUP_DIR/postfix-$DATE.tar.gz /etc/postfix/

# DKIM
tar czf $BACKUP_DIR/dkim-$DATE.tar.gz /etc/opendkim/
```



```
# SSL
tar czf $BACKUP_DIR/ssl-$DATE.tar.gz /etc/letsencrypt/
echo "✅ Backup completed: $DATE"
```

```
# Rendre exécutables
sudo chmod +x /usr/local/bin/backup-postfix.sh

# Tester
sudo /usr/local/bin/backup-postfix.sh

# Vérifier
ls -lh /backup/postfix/
```



```
# Automatiser avec cron
sudo crontab -e

# Ajouter :
0 2 * * * /usr/local/bin/backup-postfix.sh

# Tester la restauration
sudo tar xzf /backup/postfix/postfix-XXXXXX.tar.gz -C /
sudo systemctl reload postfix
```



Projet final : Serveur mail complet

Objectif

Créer un serveur mail complet et fonctionnel de A à Z.



Cahier des charges

Votre serveur mail doit :

1. Envoyer et recevoir des emails
2. Avoir un DNS correct (A, MX, PTR)
3. Avoir SPF, DKIM et DMARC configurés
4. Être sécurisé avec TLS
5. Avoir des protections anti-spam
6. Avoir des alias et domaines virtuels
7. Être sauvegardé quotidiennement
8. Être surveillé (logs, queue)

✓ Critères de validation

✉ Fonctionnalité

- Email à Gmail passe (inbox, pas spam)
- Email depuis Gmail arrive
- Score sur <https://www.mail-tester.com/> > 8/10

Sécurité

- TLS 1.2+ uniquement
- SPF, DKIM, DMARC configurés
- RBL actifs
- Authentification SASL (si port 587)

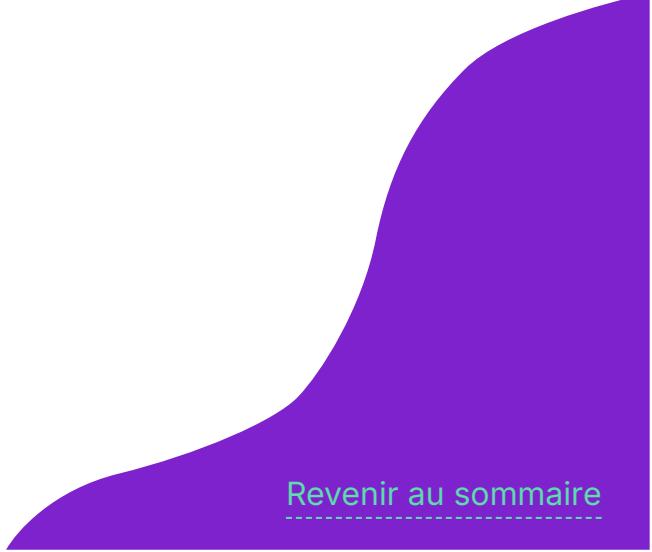
Administration

- Sauvegardes quotidiennes automatiques
- Monitoring de la queue
- Logs accessibles et analysables



Checklist

- Postfix installé et configuré
- DNS : A, MX, PTR
- SPF publié
- DKIM signé et publié
- DMARC configuré
- TLS activé (Let's Encrypt)



[Revenir au sommaire](#)

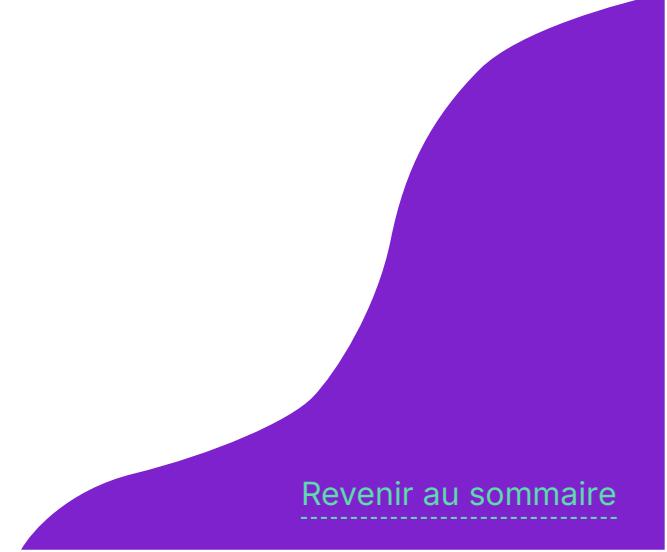
- Restrictions anti-spam
- RBL configurés
- Alias système configurés
- Domaine virtuel fonctionnel
- Sauvegarde automatique
- Test d'envoi/réception réussi

Félicitations !

Vous avez terminé les exercices de la formation **Initiation Postfix** !

Vous êtes maintenant capable de :

- Installer et configurer Postfix
- Sécuriser un serveur mail
- Gérer les domaines et alias
- Protéger contre le spam
- Surveiller et sauvegarder



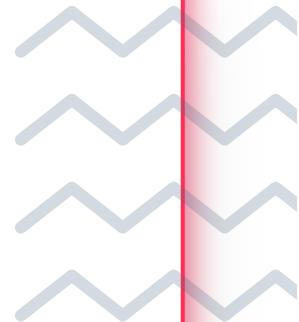
[Revenir au sommaire](#)

Prochaine étape

DIRECTION le **QCM de validation** pour tester vos connaissances ! 

Module suivant : QCM initiation →





QCM de Validation - Initiation

Testez vos connaissances Postfix !

Instructions

Format : QCM avec une seule bonne réponse par question

- **Durée :** 30 minutes
- **Score minimum :** 70% (21/30)
- **Consignes :** Lisez attentivement chaque question avant de répondre



Questions – Concepts de base

Question 1

Postfix est un :

- A) MUA (Mail User Agent)
- B) MTA (Mail Transfer Agent)
- C) MDA (Mail Delivery Agent)
- D) POP3 server

Question 2

Le fichier de configuration principal de Postfix est :

- A) /etc/postfix/postfix.conf
- B) /etc/postfix/main.cf
- C) /etc/postfix/config.cf
- D) /etc/mail/postfix.conf

Question 3

Quel processus Postfix gère les files d'attente ?

- A) smtpd
- B) smtp
- C) qmgr
- D) cleanup

Question 4

La commande pour recharger la configuration Postfix sans interruption est :

- A) systemctl restart postfix
- B) systemctl reload postfix
- C) postfix restart
- D) postfix refresh

Question 5

L'enregistrement DNS obligatoire pour un serveur mail est :

- A) A
- B) CNAME
- C) MX
- D) TXT

Questions - Configuration

Question 6

Pour accepter les emails pour `example.com`, il faut configurer :

- A) `myorigin`
- B) `mydestination`
- C) `relay_domains`
- D) `virtual_alias_domains`

Question 7

Pour éviter que votre serveur devienne un open relay, il faut absolument avoir :

- A) smtpd_tls_security_level = encrypt
- B) reject_unauth_destination
- C) smtpd_helo_required = yes
- D) disable_vrfy_command = yes



Question 8

Pour compiler le fichier `/etc/postfix/virtual` , on utilise :

- A) `postconf /etc/postfix/virtual`
- B) `postmap /etc/postfix/virtual`
- C) `newaliases`
- D) `postfix reload`



Question 9

Le format de mailbox recommandé en 2025 est :

- A) mbox
- B) Maildir
- C) mh
- D) maildir++

Question 10

Pour masquer la version de Postfix dans la bannière, on configure :

- A) smtpd_banner = \$myhostname ESMTP
- B) hide_version = yes
- C) smtpd_show_version = no
- D) banner_anonymize = yes



Questions - Files d'attente

Question 11

La commande pour voir la file d'attente est :

- A) postqueue
- B) mailq
- C) queuelist
- D) showqueue



Question 12

Pour supprimer TOUS les messages de la queue, on utilise :

- A) postsuper -d ALL
- B) postqueue -d ALL
- C) mailq --delete-all
- D) postfix flush



Question 13

Un message avec le statut `deferred` signifie :

- A) Le message est supprimé
- B) Le message est en erreur temporaire
- C) Le message est en erreur permanente
- D) Le message est en cours d'envoi

Question 14

Pour forcer l'envoi immédiat de tous les messages en queue :

- A) postsuper -f
- B) postqueue -f
- C) mailq -send
- D) postfix send

Questions - Sécurité

Question 15

SPF permet de :

- A) Signer cryptographiquement les emails
- B) Lister les serveurs autorisés à envoyer pour un domaine
- C) Chiffrer les communications SMTP
- D) Authentifier les utilisateurs

Question 16

DKIM utilise :

- A) Un enregistrement MX
- B) Une signature cryptographique
- C) Un certificat SSL
- D) Un mot de passe

Question 17

Dans un enregistrement SPF, -all signifie :

- A) Tout est autorisé
- B) Tout est rejeté sauf les serveurs listés
- C) Pas de SPF
- D) Mode test

Question 18

La version TLS minimum recommandée en 2025 est :

- A) TLS 1.0
- B) TLS 1.1
- C) TLS 1.2
- D) TLS 1.3



Question 19

Le port standard pour la submission (envoi client) est :

- A) 25
- B) 465
- C) 587
- D) 993

Question 20

Pour activer TLS en mode opportuniste (si disponible) :

- A) smtpd_tls_security_level = none
- B) smtpd_tls_security_level = may
- C) smtpd_tls_security_level = encrypt
- D) smtpd_tls_security_level = mandatory

Questions - Anti-spam

Question 21

Un RBL (Real-time Blackhole List) permet de :

- A) Bloquer les IPs connues pour envoyer du spam
- B) Lister les emails légitimes
- C) Chiffrer les emails
- D) Authentifier les utilisateurs

Question 22

Le RBL le plus utilisé est :

- A) spamcop.net
- B) zen.spamhaus.org
- C) blacklist.mail.com
- D) rbl.google.com

Question 23

Le greylisting fonctionne sur le principe que :

- A) Les spammeurs ne réessaient pas
- B) Les emails sont marqués comme spam
- C) Les IPs sont blacklistées
- D) Les emails sont chiffrés

Question 24

Pour rejeter les HELO invalides, on configure :

- A) smtpd_helo_required = yes + reject_invalid_helo_hostname
- B) smtpd_reject_helo = yes
- C) smtpd_check_helo = strict
- D) helo_restrictions = reject_all

Questions - Administration

Question 25

Les logs Postfix se trouvent par défaut dans :

- A) /var/log/postfix.log (Ubuntu/Debian)
- B) /var/log/mail.log (Ubuntu/Debian)
- C) /var/log/messages (Ubuntu/Debian)
- D) /etc/postfix/logs/

Question 26

Pour suivre les logs en temps réel, on utilise :

- A) cat /var/log/mail.log
- B) less /var/log/mail.log
- C) tail -f /var/log/mail.log
- D) head /var/log/mail.log

Question 27

Pour vérifier la syntaxe du fichier `main.cf`, on utilise :

- A) postfix test
- B) postconf -t
- C) postfix check
- D) postconf --verify

Question 28

Un backup de Postfix doit au minimum inclure :

- A) /var/spool/postfix/ uniquement
- B) /etc/postfix/ uniquement
- C) /etc/postfix/ et /etc/opendkim/
- D) /var/log/mail.log



Question 29

Pour automatiser une sauvegarde quotidienne à 2h du matin, on ajoute dans cron :

- A) 0 2 * * * /backup.sh
- B) 2 0 * * * /backup.sh
- C) * 2 * * * /backup.sh
- D) 0 2 1 * * /backup.sh

Question 30

La règle 3-2-1 pour les sauvegardes signifie :

- A) 3 serveurs, 2 backups, 1 admin
- B) 3 copies, 2 supports, 1 hors site
- C) 3 jours, 2 semaines, 1 mois
- D) 3 fichiers, 2 disques, 1 cloud

Réponses - QCM Initiation (1/3)

Q1 : B - MTA (Mail Transfer Agent) - Postfix achemine les emails entre serveurs.

Q2 : B - /etc/postfix/main.cf - Fichier de configuration centrale.

Q3 : C - qmgr - Le Queue Manager gère toutes les files d'attente.

Q4 : B - systemctl reload postfix - Reload sans couper les connexions.

Q5 : C - MX - Enregistrement MX indique le serveur mail du domaine.

Q6 : B - mydestination - Liste les domaines pour livraison locale.

Q7 : B - reject_unauth_destination - Empêche le relais non autorisé.

Q8 : B - postmap /etc/postfix/virtual - Compile les tables.

Q9 : B - Maildir - Plus sûr et performant que mbox.

Q10 : A - smtpd_banner = \$myhostname ESMTP - Sans \$mail_name ou \$mail_version .



Réponses - QCM Initiation (2/3)

Q11 : B - `mailq` (ou `postqueue -p`) - Affiche la queue.

Q12 : A - `postsuper -d ALL` - Gère les opérations sur la queue.

Q13 : B - Erreur temporaire - `deferred` sera réessayé.

Q14 : B - `postqueue -f` - Force le traitement immédiat.

Q15 : B - Lister les serveurs autorisés - SPF = enregistrement DNS des IPs autorisées.

Q16 : B - Signature cryptographique - DKIM signe avec clé privée.

Q17 : B - Tout rejeté sauf serveurs listés - `-all strict (FAIL)`, `~all permissif (SOFTFAIL)`.

Q18 : C - TLS 1.2 - TLS 1.2 minimum, idéalement TLS 1.3.

Q19 : C - 587 - Port submission avec STARTTLS obligatoire.

Q20 : B - `smtpd_tls_security_level = may` - TLS si client supporte.

^ ^