Configurer un relais SMTP avec Postfix et Resend (RELAIS SMTP SANS IP PUBLIQUE)

Principe

Application (exemple Documenso) ⇒ Postfix (port 25) ⇒ Script Python (Communication via LPIPE) ⇒ Microsoft Graph API

Préparation Azure AD (OAuth2)

- création d'une **Inscription d'applications** Entra ID :
  - Portail Azure ⇒ Entra ID
  - Inscription d'applications ⇒ Nouvelle inscription
  - Nom : smtp2graph-relay
  - Locataire unique seulement
  - S'incrire
- Récupérer :
  - Tenant ID
  - Client ID
  - Ajouter un secret Client (dans Certificates & Secrets)
- Ajouter la permission Microsoft Graph :
  - Autorisations d'application (et non Autorisations déléguées) :Mail.Send
- Grant admin consent.
- Adresse email 0365 utilisée pour l'envoi afin que l'app puisse avoir le droit d'envoyer au nom de ce compte.

Installation des prérequis

- conteneur LXC : 2 Gio RAM ; 2 coeurs ; DD de 20 Gio
- modifier le fichier **/etc/apt/sources.list.d/debian.sources** pour avoir ce contenu (http://security.debian.org trixie-security remplacé par http://deb.debian.org/debian-security)

```
Types: deb
URIs: http://deb.debian.org/debian-security
Suites: trixie-security
Components: contrib main
Signed-By: /usr/share/keyrings/debian-archive-keyring.gpg

Types: deb
URIs: http://deb.debian.org/debian
Suites: trixie trixie-updates
Components: contrib main
Signed-By: /usr/share/keyrings/debian-archive-keyring.gpg
```

- . installer les paquets pour le script

```
apt install mailutils libsasl2-modules
apt install python3-pip
```

Script Python

- créer le fichier **/usr/local/bin/graph_sendmail.py**

```
#!/usr/bin/env python3
import sys
import email
import requests
import json
import base64
import traceback
from email.utils import getaddresses


# ------------------------------------------------
```

```
# CONFIG MICROSOFT GRAPH
# ------------------------------------------------
TENANT = "TON_TENANT_ID"
CLIENT_ID = "TON_CLIENT_ID"
CLIENT_SECRET = "TON_CLIENT_SECRET"
FROM_ADDR = "ton.adresse@tondomaine.com"

DEBUG_LOG = "/tmp/graph_debug.log"

def debug(msg):
    with open(DEBUG_LOG, "a", encoding="utf-8") as f:
        f.write(msg + "\n")


def clean_address_list(field_value):
    """
    Extrait correctement toutes les adresses
    même si Documenso génère des listes complexes.
    """
    if not field_value:
        return []

    parsed = getaddresses([field_value])
    cleaned = []

    for name, addr in parsed:
        addr = addr.replace("<", "").replace(">", "").strip()
        if addr:
            cleaned.append(addr)

    return cleaned


def flatten_body(part, text_body, html_body, attachments, inline_images):
    """
    Parcourt récursivement l'arbre MIME.
    """
    ctype = part.get_content_type()
    disp = str(part.get("Content-Disposition", "")).lower()

    # 1. Si multipart → parcourir les sous-parties
    if part.is_multipart():
        for subpart in part.get_payload():
            flatten_body(subpart, text_body, html_body, attachments, inline_images)
        return

    # 2. Corps texte
    if ctype == "text/plain" and "attachment" not in disp:
        payload = part.get_payload(decode=True)
        if payload:
            text_body.append(payload.decode("utf-8", errors="ignore"))
        return

    # 3. Corps HTML
    if ctype == "text/html" and "attachment" not in disp:
        payload = part.get_payload(decode=True)
        if payload:
            html_body.append(payload.decode("utf-8", errors="ignore"))
        return

    # 4. Images inline (multipart/related)
    if "inline" in disp and ctype.startswith("image/"):
        cid = part.get("Content-ID", "").strip("<>")
        payload = part.get_payload(decode=True)
        if payload and cid:
            inline_images.append({
                "@odata.type": "#microsoft.graph.fileAttachment",
                "name": cid,
                "contentId": cid,
                "isInline": True,
```

```
                "contentBytes": base64.b64encode(payload).decode()
            })
        return

    # 5. Attachments
    if "attachment" in disp or part.get_filename():
        filename = part.get_filename()
        payload = part.get_payload(decode=True)
        if filename and payload:
            attachments.append({
                "@odata.type": "#microsoft.graph.fileAttachment",
                "name": filename,
                "contentBytes": base64.b64encode(payload).decode()
            })
        return


# ---------------------------------------------------
# MAIN
# ---------------------------------------------------
try:
    raw = sys.stdin.read()
    msg = email.message_from_string(raw)

    debug("=== Nouveau mail reçu ===")

    subject = msg.get("Subject", "(no subject)")
    debug("Sujet: " + subject)

    # Extraction des adresses MIME (Documenso friendly)
    to_list = clean_address_list(msg.get("To"))
    cc_list = clean_address_list(msg.get("Cc"))
    bcc_list = clean_address_list(msg.get("Bcc"))
    reply_to_list = clean_address_list(msg.get("Reply-To"))

    debug("To: " + str(to_list))
    debug("Cc: " + str(cc_list))
    debug("Bcc: " + str(bcc_list))
    debug("Reply-To: " + str(reply_to_list))

    # Corps du message
    text_body = []
    html_body = []
    attachments = []
    inline_images = []

    flatten_body(msg, text_body, html_body, attachments, inline_images)

    # Déterminer le corps principal HTML
    final_html = ""

    if html_body:
        final_html = "\n".join(html_body)
    elif text_body:
        final_html = "<pre>" + "\n".join(text_body) + "</pre>"
    else:
        final_html = "<p>(vide)</p>"

    debug("Corps HTML (200 chars): " + final_html[:200])
    debug("Attachments: " + str(len(attachments)))
    debug("Inline images: " + str(len(inline_images)))

    # ---------------------------------------------------
    # TOKEN OAUTH2
    # ---------------------------------------------------
    debug("Before OAuth token request")

    token_res = requests.post(
        f"https://login.microsoftonline.com/{TENANT}/oauth2/v2.0/token",
        data={
```

```
            "client_id": CLIENT_ID,
            "scope": "https://graph.microsoft.com/.default",
            "client_secret": CLIENT_SECRET,
            "grant_type": "client_credentials"
        }
    )

    debug("Token response: " + token_res.text)

    token_json = token_res.json()
    if "access_token" not in token_json:
        raise Exception("OAuth2 failed: " + token_res.text)

    token = token_json["access_token"]

    # ------------------------------------------------
    # CONSTRUCTION JSON GRAPH
    # ------------------------------------------------
    def addr_obj(addr):
        return {"emailAddress": {"address": addr}}

    message = {
        "subject": subject,
        "body": {
            "contentType": "HTML",
            "content": final_html
        },
        "toRecipients": [addr_obj(a) for a in to_list],
        "ccRecipients": [addr_obj(a) for a in cc_list],
        "bccRecipients": [addr_obj(a) for a in bcc_list],
        "attachments": attachments + inline_images
    }

    if reply_to_list:
        message["replyTo"] = [addr_obj(a) for a in reply_to_list]

    mail_json = {
        "message": message,
        "saveToSentItems": True
    }

    debug("Mail JSON ready")

    # ------------------------------------------------
    # ENVOI VIA GRAPH API
    # ------------------------------------------------
    graph_res = requests.post(
        f"https://graph.microsoft.com/v1.0/users/{FROM_ADDR}/sendMail",
        headers={
            "Authorization": f"Bearer {token}",
            "Content-Type": "application/json"
        },
        data=json.dumps(mail_json)
    )

    debug("Graph sendMail response: " + graph_res.text)
    debug("Status: " + str(graph_res.status_code))

except Exception as e:
    debug("SCRIPT ERROR: " + str(e))
    debug(traceback.format_exc())
```

- Rendre le script exécutable :

```
chmod 755 /usr/local/bin/graph_sendmail.py
chown root:root /usr/local/bin/graph_sendmail.py
```

Configurer Postfix

utiliser LPIPE pour appeler le script

- Créer le fichier **/etc/postfix/transport** :

```
*     graph:
```

- compiler

```
postmap /etc/postfix/transport
```

- vérifier

```
postmap -s /etc/postfix/transport
```

- vérifier que le transport existe

```
#postconf -M
...
graph unix - n n - - pipe
```

- Configurer Postfix en modifiant le fichier /etc/postfix/main.cf pour ajouter :

```
#myhostname = postfix-relay.lan
#mydomain = lan
#mydestination =
#relayhost =

# Postfix doit écouter sur toutes les interfaces
inet_interfaces = all

# Autoriser l'IP des serveurs client
mynetworks = 127.0.0.0/8 9.9.9.0/24

# Postfix doit accepter les mails :
smtpd_recipient_restrictions = permit_mynetworks, reject_unauth_destination


transport_maps = hash:/etc/postfix/transport

lpipe_destination_recipient_limit = 1
```

- Config /etc/postfix/master.cf pour appeler le script Python en ajoutant à la fin du fichier :

```
graph   unix  -   n   n   -   -   pipe
  flags=Fq.
  user=nobody
  argv=/scripts/graph_sendmail.py
```

- relancer Postfix :

```
systemctl restart postfix
```

Tester

- envoyer un message

```
echo "hello" | mail -s "test mime" -A document.pdf charles@gmail.xxx
```

- vérifier dans les logs générés par le script qu'il n'y a pas d'erreur (le statut doit être 202) :

```
cat /tmp/graph_debug.log
```

- vérifier dans les logs de Potsfix qu'il n'y a pas d'erreur :

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