```yaml
services:
  traefik:
    image: "traefik"
    restart: always
    command:
      - "--api.insecure=true"
      - "--providers.docker=true"
      - "--providers.docker.exposedbydefault=false"
      - "--entrypoints.web.address=:80"
      - "--entrypoints.websecure.address=:443"
      - "--certificatesresolvers.mytlschallenge.acme.tlschallenge=true"
      - "--certificatesresolvers.mytlschallenge.acme.email=tu_email@example.com"
      - "--certificatesresolvers.mytlschallenge.acme.storage=/letsencrypt/acme.json"
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - traefik_data:/letsencrypt
      - /var/run/docker.sock:/var/run/docker.sock:ro

  n8n:
    image: docker.n8n.io/n8nio/n8n
    restart: always
    environment:
      - N8N_HOST=n8n.example.com
      - N8N_PROTOCOL=https
      - WEBHOOK_URL=https://n8n.example.com/
      - TZ=Atlantic/Canary
      - N8N_ENFORCE_SETTINGS_FILE_PERMISSIONS=true
      - N8N_RUNNERS_ENABLED=true
      - NODE_ENV=production
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.n8n.rule=Host(`n8n.example.com`)"
      - "traefik.http.routers.n8n.entrypoints=websecure"
      - "traefik.http.routers.n8n.tls.certresolver=mytlschallenge"
      - "traefik.http.middlewares.n8n-redirect.redirectscheme.scheme=https"
      - "traefik.http.routers.n8n-http.rule=Host(`n8n.example.com`)"
      - "traefik.http.routers.n8n-http.entrypoints=web"
      - "traefik.http.routers.n8n-http.middlewares=n8n-redirect"
    volumes:
      - n8n_data:/home/node/.n8n
      - ./local-files:/files

volumes:
  n8n_data:
  traefik_data:
```

Claro, aquí tienes la documentación actualizada para tu nuevo archivo `docker-compose.yml`, que ahora incluye **Traefik** como proxy inverso y gestor de certificados SSL. La estructura es más robusta y está orientada a un entorno de producción con HTTPS.

Función del Archivo `docker-compose.yml`

Este archivo define la configuración para dos servicios: `traefik` y `n8n`. La principal diferencia con tu configuración anterior es la inclusión de Traefik, que se encarga de dirigir el tráfico web y automatizar la gestión de certificados SSL para asegurar tu instancia de n8n.

Servicio `traefik`

Traefik actúa como un **proxy inverso** dinámico. Su configuración permite manejar las peticiones que llegan a tu servidor y dirigirlas al contenedor de n8n, además de automatizar la obtención de certificados SSL con Let's Encrypt.

- `image: "traefik"`: Utiliza la imagen oficial de Traefik.
- `command:`: Define los parámetros de inicio de Traefik.
  - `--api.insecure=true`: Expone la API de Traefik sin seguridad, útil para monitoreo.
  - `--providers.docker=true`: Permite a Traefik descubrir servicios de Docker de forma automática.

- `--providers.docker.exposedbydefault=false`: Evita que Traefik gestione contenedores por defecto. Solo gestionará aquellos con etiquetas específicas.
- `--entrypoints.web.address=:80` y `--entrypoints.websecure.address=:443`: Definen los puntos de entrada para el tráfico HTTP y HTTPS, respectivamente.
- `--certificatesresolvers.mytlschallenge...`: Configuran el gestor de certificados de Let's Encrypt para obtener certificados SSL. **Asegúrate de cambiar `tu_email@example.com` por tu dirección de correo electrónico real.**
- `ports: - "80:80" - "443:443"`: Mapea los puertos estándar de tráfico web (HTTP y HTTPS) de tu máquina anfitriona a Traefik.
- `volumes:`: Define la persistencia de los datos de Traefik.
  - `traefik_data:/letsencrypt`: Un volumen con nombre que almacena los certificados SSL.
  - `/var/run/docker.sock:/var/run/docker.sock:ro`: Permite que Traefik se comunique con el demonio de Docker para descubrir y configurar dinámicamente los servicios.

Servicio `n8n`

Este contenedor sigue ejecutando n8n, pero ahora su acceso está mediado por Traefik.

- `image: docker.n8n.io/n8nio/n8n`: La imagen de n8n.
- `environment:`: Variables de entorno de n8n.
  - `N8N_HOST`, `N8N_PROTOCOL`, `WEBHOOK_URL`: Estas variables están configuradas para usar **HTTPS** y tu dominio específico (`n8n.example.com`), lo cual es crucial para que Traefik y Let's Encrypt funcionen correctamente.
  - `N8N_ENFORCE_SETTINGS_FILE_PERMISSIONS=true`: Una buena práctica de seguridad.
  - `N8N_RUNNERS_ENABLED=true`: Habilita la ejecución de flujos de trabajo en n8n.
- `labels:`: Estas etiquetas son la clave de la integración. Le indican a Traefik cómo manejar este contenedor.
  - `traefik.enable=true`: Le dice a Traefik que gestione este contenedor.
  - `traefik.http.routers.n8n.rule=Host(...)`: Define una regla para enrutar el tráfico basado en el nombre del host.
  - `traefik.http.routers.n8n.entrypoints=websecure`: Configura el enrutador para que use el punto de entrada HTTPS.
  - `traefik.http.routers.n8n.tls.certresolver=mytlschallenge`: Le indica a Traefik que use el gestor de certificados que configuraste para obtener un certificado SSL.
  - Las demás etiquetas (`n8n-http...`) configuran una redirección automática de HTTP a HTTPS, una práctica estándar de seguridad.

Cómo Montar y Ejecutar n8n con Traefik

Para poner en marcha esta nueva configuración, sigue estos pasos:

1. **Requisitos**: Asegúrate de tener **Docker** y **Docker Compose** instalados. Además, tu servidor debe tener los puertos **80 y 443 abiertos** y un **nombre de dominio** (`n8n.example.com`) que apunte a la dirección IP de tu servidor. 🌐
2. **Crea el archivo `docker-compose.yml`**: En la carpeta raíz de tu proyecto, crea este archivo y una carpeta llamada `local-files`.
3. **Copia y pega el nuevo código**: Pega el código actualizado en el archivo `docker-compose.yml`. **No olvides cambiar `tu_email@example.com` por tu correo.**
4. **Ejecuta el comando**: Abre una terminal en la misma carpeta y ejecuta:

   ```
   docker compose up -d
   ```

   Este comando creará y ejecutará ambos contenedores (`traefik` y `n8n`) en segundo plano.
5. **Verifica la instalación**: Traefik tardará unos segundos en obtener el certificado SSL. Una vez que termine, podrás acceder a tu instancia de n8n de forma segura a través de `https://n8n.example.com`.

Para detener la instancia de n8n y Traefik, simplemente ejecuta `docker compose down` en la misma carpeta.