

Comment installer le réseau social Mastodon sur Debian 12

Mastodon est un réseau social gratuit, décentralisé et open source. Il a été créé comme alternative à Twitter. Tout comme Twitter, les gens peuvent se suivre et publier des messages, des images et vidéos. Mais contrairement à Twitter, il n'existe pas de magasin central ni d'autorité pour le contenu.

Au lieu de cela, Mastodon opère sur des milliers de serveurs différents, chacun géré par différents membres de la communauté. Les utilisateurs inscrits sur un serveur peuvent facilement se connecter aux utilisateurs de l'autre réseau et se suivre entre les instances.

N'importe qui peut installer son instance d'un serveur Mastodon. Ce tutoriel va vous apprendre comment configurer votre instance de Mastodon sur un serveur avec Debian 12 en utilisant Docker. Docker facilite l'installation de Mastodon en contenant tous les packages et services requis dans des conteneurs.

Conditions préalables

- Un serveur exécutant Debian 12 avec un minimum de 2 cœurs de processeur et 2 Go de mémoire. Vous devrez mettre à niveau le serveur selon les exigences.
- Un utilisateur non root avec les privilèges sudo.
- Un nom de domaine complet (FQDN) pointant vers votre serveur. Pour nos besoins, nous utiliserons mastodon.example.com comme nom de domaine.
- Mastodon envoie des notifications par e-mail aux utilisateurs. Nous vous recommandons d'utiliser un service de courrier transactionnel tels que Mailgun, SendGrid, Amazon SES ou Sparkpost. Les instructions contenues dans le guide utiliseront Amazon SES.
- Assurez-vous que tout est mis à jour.

```
$ sudo apt update
```

- Installez les packages utilitaires de base. Certains d'entre eux sont peut-être déjà installés.

```
$ sudo apt install curl wget nano software-properties-common dirmngr apt-transport-https ca-certificates lsb-release debian-archive-keyring gnupg2 ufw unzip -y
```

Étape 1 - Configurer le pare-feu

La première étape consiste à configurer le pare-feu. Debian est livré par défaut avec ufw (Uncomplicated Firewall).

Vérifiez si le pare-feu est en cours d'exécution.

```
$ sudo ufw status
```

Vous devriez obtenir le résultat suivant.

```
Status: inactive
```

Autorisez le port SSH afin que le pare-feu ne rompe pas la connexion actuelle en l'activant.

```
$ sudo ufw allow OpenSSH
```

Autorisez également les ports HTTP et HTTPS.

```
$ sudo ufw allow http
$ sudo ufw allow https
```

Activer le pare-feu

```
$ sudo ufw enable
Command may disrupt existing ssh connections. Proceed with operation (y/n)? y
Firewall is active and enabled on system startup
```

Vérifiez à nouveau l'état du pare-feu.

```
$ sudo ufw status
```

Vous devriez voir une sortie similaire.

```
Status: active

To Action From
--
OpenSSH ALLOW Anywhere
90/tcp ALLOW Anywhere
443 ALLOW Anywhere
OpenSSH (v6) ALLOW Anywhere (v6)
90/tcp (v6) ALLOW Anywhere (v6)
443 (v6) ALLOW Anywhere (v6)
```

Étape 2 - Installer Docker et Docker Compose

Debian 12 est livré avec une ancienne version de Docker. Pour installer la dernière version, importez d'abord la clé Docker GPG.

```
$ sudo install -m 0755 -d /etc/apt/keysrings
$ curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg --dearmor -o /etc/apt/keysrings/docker.gpg
$ sudo chmod a+r /etc/apt/keysrings/docker.gpg
```

Créez le fichier de référentiel Docker.

```
$ echo \
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keysrings/docker.gpg] https://download.docker.com/linux/debian \
$(. /etc/os-release && echo "VERSION_CODENAME") stable" | \
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

Mettez à jour la liste des référentiels système.

```
$ sudo apt update
```

Installez la dernière version de Docker.

```
$ sudo apt install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

Vérifiez qu'il est en cours d'exécution.

```
$ sudo systemctl status docker
? docker.service - Docker Application Container Engine
Loaded: loaded (/lib/systemd/system/docker.service; enabled; preset: enabled)
Active: active (running) since Mon 2024-01-01 09:00:14 UTC; 17s ago
TriggeredBy: ? docker.socket
Docs: https://docs.docker.com
Main PID: 1839 (dockerd)
Tasks: 9
Memory: 27.6M
CPU: 598ms
CGroup: /system.slice/docker.service
?1839 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
```

Par défaut, Docker nécessite les privilèges root. Si vous souhaitez éviter d'utiliser `sudo` chaque fois que vous exécutez la commande `docker`, ajoutez votre nom d'utilisateur au groupe `docker`.

```
$ sudo usermod -aG docker $(whoami)
```

Vous devez vous déconnecter du serveur et vous reconnecter en tant que même utilisateur pour activer cette modification ou utiliser la commande suivante.

```
$ su - ${USER}
```

Confirmez que votre utilisateur est ajouté au groupe Docker.

```
$ groups  
navjot sudo users docker
```

Étape 3 - Préparation de l'installation

La limite par défaut du nombre de mmaps est très faible pour Elasticsearch. Exécutez la commande suivante pour vérifier la valeur par défaut.

```
$ sudo sysctl vm.max_map_count
```

Vous obtiendrez le résultat suivant.

```
vm.max_map_count = 65530
```

Augmentez la valeur à l'aide des commandes suivantes.

```
$ echo "vm.max_map_count=262144" | sudo tee /etc/sysctl.d/90-max_map_count.conf  
vm.max_map_count=262144  
$ sudo sysctl --load /etc/sysctl.d/90-max_map_count.conf  
vm.max_map_count=262144
```

Étape 4 - Installer Mastodon

Créer des répertoires et définir les propriétés

Créer des répertoires pour Mastodon et les services associés.

```
$ sudo mkdir -p /opt/mastodon/database/{postgresql,pgbackups,redis,elasticsearch}  
$ sudo mkdir -p /opt/mastodon/web/{public,system}  
$ sudo mkdir -p /opt/mastodon/branding
```

Définissez les propriétés appropriées pour les répertoires Elasticsearch, Web et de sauvegarde.

```
$ sudo chown 991:991 /opt/mastodon/web/{public,system}  
$ sudo chown 1000 /opt/mastodon/database/elasticsearch  
$ sudo chown 70:70 /opt/mastodon/database/pgbackups
```

Basculez vers le répertoire Mastodon.

```
$ cd /opt/mastodon
```

Créer un environnement et Docker composer des fichiers

Créer des fichiers d'environnement pour l'application et la base de données.

```
$ sudo touch application.env database.env
```

Créer et ouvrez le fichier de composition Docker pour le modifier.

```
$ sudo nano docker-compose.yml
```

Collez-y le code suivant.

```
services:  
  postgresql:  
    image: postgres:16-alpine  
    env_file: database.env  
    restart: always  
    shm_size: 512mb  
    healthcheck:  
      test: ['CMD', 'pg_isready', '-U', 'postgres']  
    volumes:  
      - postgresql:/var/lib/postgresql/data  
      - pgbackups:/backups  
    networks:  
      - internal_network  
  redis:  
    image: redis:7-alpine  
    restart: always  
    healthcheck:  
      test: ['CMD', 'redis-cli', 'ping']  
    volumes:  
      - redis:/data  
    networks:  
      - internal_network  
  redis-volatile:  
    image: redis:7-alpine  
    restart: always  
    healthcheck:  
      test: ['CMD', 'redis-cli', 'ping']  
    networks:  
      - internal_network  
  elasticsearch:  
    image: docker.elastic.co/elasticsearch/elasticsearch:7.17.16  
    restart: always  
    env_file: database.env  
    environment:  
      - cluster.name=elasticsearch-mastodon  
      - discovery.type=single-node  
      - bootstrap.memory_lock=true  
      - xpack.security.enabled=true  
      - ingest.geoip.downloader.enabled=false  
      - "ES_JAVA_OPTS=-Xms512m -Xmx512m -Des.enforce.bootstrap.checks=true"  
      - xpack.license.self_generated.type=basic  
      - xpack.watcher.enabled=false  
      - xpack.graph.enabled=false  
      - xpack.ml.enabled=false  
      - thread_pool.write.queue_size=1000  
    ulimits:  
      memlock:  
        soft: -1  
        hard: -1  
      nofile:  
        soft: 65536  
        hard: 65536  
    healthcheck:  
      test: ['CMD-SHELL', "nc -z elasticsearch 9200"]  
    volumes:  
      - elasticsearch:/usr/share/elasticsearch/data  
    networks:  
      - internal_network  
    ports:  
      - "127.0.0.1:9200:9200"  
  website:  
    image: tootsuite/mastodon:v4.2.3  
    env_file:  
      - application.env  
      - database.env  
    command: bash -c "rm -f /mastodon/tmp/pids/server.pid; bundle exec rails s -p 3000"  
    restart: always  
    depends_on:  
      - postgresql  
      - redis  
      - redis-volatile
```

```

- elasticsearch
ports:
- '127.0.0.1:3000:3000'
networks:
- internal_network
- external_network
healthcheck:
test: ['CMD-SHELL', 'wget -q --spider --proxy=off localhost:3000/health || exit 1']
volumes:
- uploads:/mastodon/public/system

shell:
image: tootsuite/mastodon:v4.2.3
env_file:
- application.env
- database.env
command: /bin/bash
restart: "no"
networks:
- internal_network
- external_network
volumes:
- uploads:/mastodon/public/system
- static:/static

streaming:
image: tootsuite/mastodon:v4.2.3
env_file:
- application.env
- database.env
command: node ./streaming
restart: always
depends_on:
- postgresql
- redis
- redis-volatile
- elasticsearch
ports:
- '127.0.0.1:4000:4000'
networks:
- internal_network
- external_network
healthcheck:
test: ['CMD-SHELL', 'wget -q --spider --proxy=off localhost:4000/api/v1/streaming/health || exit 1']

sidekiq:
image: tootsuite/mastodon:v4.2.3
env_file:
- application.env
- database.env
command: bundle exec sidekiq
restart: always
depends_on:
- postgresql
- redis
- redis-volatile
- website
networks:
- internal_network
- external_network
healthcheck:
test: ['CMD-SHELL', "ps aux | grep '[s]idekiq\ 6' || false"]
volumes:
- uploads:/mastodon/public/system

networks:
external_network:
internal_network:
internal: true

volumes:
postgresql:
driver_opts:
type: none
device: /opt/mastodon/database/postgresql
o: bind
pgbackups:
driver_opts:
type: none
device: /opt/mastodon/database/pgbackups
o: bind
redis:
driver_opts:
type: none
device: /opt/mastodon/database/redis
o: bind
elasticsearch:
driver_opts:
type: none
device: /opt/mastodon/database/elasticsearch
o: bind
uploads:
driver_opts:
type: none
device: /opt/mastodon/web/system
o: bind
static:
driver_opts:
type: none
device: /opt/mastodon/web/public
o: bind

```

Enregistrez le fichier en appuyant sur Ctrl + X et en entrant Y lorsque vous y êtes invité.

Au moment de la rédaction de ce tutoriel, la dernière version disponible de Mastodon est la v4.2.3. Consultez la [page des versions de Mastodon GitHub](#) et ajustez la version dans le fichier de composition Docker de manière appropriée. Nous utilisons également les dernières versions de PostgreSQL et Redis. Vous pouvez les ajuster selon vos besoins. Nous utilisons actuellement [Elasticsearch 7.17.16](#).

Créer des secrets d'application

L'étape suivante consiste à créer des valeurs secrètes d'application.

Générez les valeurs SECRET_KEY_BASE et OTP_SECRET en exécutant deux fois la commande suivante. La première fois prendra un certain temps car elle extraira les images.

```

$ docker compose run --rm shell bundle exec rake secret
349623cd049e3b856f6848638146e459857862b908ed387bbef372a30d9bd7c604fc4de5338addc86bd369a99d38ef59bacfa28e02a1750f7094ea6ede05457b8

```

Vous pouvez également utiliser l'utilitaire openssl pour la même chose.

```

$ openssl rand -hex 64
ae01cd7d4d4ae0182461a1345f1f2bf159658a27339ffafe7d356bef9ee844fa015ab2e72a608f236bd8e3f9b2af2dcb1d55ee5c8e43646959112c7da558244b

```

Générez les valeurs VAPID_PRIVATE_KEY et VAPID_PUBLIC_KEY à l'aide de la commande suivante.

```

$ docker compose run --rm shell bundle exec rake mastodon:webpush:generate_vapid_key

```

Vous obtiendrez un résultat similaire.

```

VAPID_PRIVATE_KEY=uz2pcCa5JdmndmMLuJ4U0gmFGvZedtsz-fB_xf4_ac=
VAPID_PUBLIC_KEY=BUJZ2h9VwFpo6dnHqyftu-2PCP-c8oL39UCmhmwAqfW_vnYp4G8neRT_om6LZUyJfHf-TBAP8S9W7qH8A=

```

Utilisez l'utilitaire openssl pour générer des mots de passe PostgreSQL et Elasticsearch.

```

$ openssl rand -hex 15
dd0bbd7a93968623e08e084a1fb7d5c
$ openssl rand -hex 15
0fb52834c991b5e296c647166185bc

```

Fichiers d'environnement Mastodon

Ouvrez le fichier application.env pour le modifier.

```

$ sudo nano application.env

```

Collez-y les lignes suivantes.

```

# environment
RAILS_ENV=production
NODE_ENV=production

```



```
# domain
LOCAL_DOMAIN=mastodon.example.com
# redirect to the first profile
SINGLE_USER_MODE=false
# do not serve static files
RAILS_SERVE_STATIC_FILES=false
# concurrency
WEB_CONCURRENCY=2
MAX_THREADS=5
# pgbouncer
#PREPARED_STATEMENTS=false
# locale
DEFAULT_LOCALE=en
# email, not used
SMTP_SERVER=email-smtp.us-west-2.amazonaws.com
SMTP_PORT=587
SMTP_LOGIN=AKIA3FIG4NVFB343PEI
SMTP_PASSWORD=AZX01WIA6JGbeZ2pwVXnyC9DhEa2nKcmXSuZbLp
SMTP_FROM_ADDRESS=noreply@nspeaks.com
# secrets
SECRET_KEY_BASE=349623c049e3b856f6848638146e459857862b908ed387b7bf372a30d9bd7c604fc4de5338addc86bd369a99d38ef59bacfa28e02a17507094ea6ede05457b8
OTP_SECRET=aa01c7d4d4fae0182461a1345f1f2bf159658a27339ffafe7d356bef9ee8d4fa015ab2e72a608f236b8e3f9b2af2dcb1d55ee5c8e43646959112c7da5582f4b
# Changing VAPID keys will break push notifications
VAPID_PRIVATE_KEY=oNe_4BEL7Tpc3iV8eMtlLegLwrzA7ifitGJ2Y0g3dUM=
VAPID_PUBLIC_KEY=BKbgmB90vlnrJg6lfq3cCHixalyPghJDKui9vm1wscvAFINNoAQL0KinoxRtLDp0UFIGK_ahUG2n4W2n4x9AUAWM=
# IP and session retention
# -----
# Make sure to modify the scheduling of ip_cleanup_scheduler in config/sidekiq.yml
# to be less than daily if you lower IP_RETENTION_PERIOD below two days (172800).
# -----
IP_RETENTION_PERIOD=2592000
SESSION_RETENTION_PERIOD=2592000
```

Enregistrez le fichier en appuyant sur Ctrl + X et en entrant Y lorsque vous y êtes invité.

Nous avons activé le service de messagerie Amazon SES. Si vous n'en avez pas besoin, vous pouvez supprimer la section. Par défaut, Mastodon conserve une adresse IP pendant 1 an, mais nous l'avons modifiée à 30 jours (2592000 secondes). Vous pouvez le modifier selon vos besoins. Assurez-vous de le conserver plus de 2 jours sinon vous devrez faire un peu plus de bricolage, ce qui sort du cadre de notre Didacticiel.

Ouvrez le fichier `database.env` pour le modifier.

```
$ sudo nano database.env
```

Collez-y les lignes suivantes.

```
# postgresql configuration
POSTGRES_USER=mastodon
POSTGRES_DB=mastodon
POSTGRES_PASSWORD=0fb52834c991b5e296c647166185bc
PGPASSWORD=0fb52834c991b5e296c647166185bc
PGPORT=5432
PGHOST=postgres
PGUSER=mastodon
# pgbouncer configuration
#POOL_MODE=transaction
#ADMIN_USERS=postgres,mastodon
#DATABASE_URL="postgres://mastodon:0fb52834c991b5e296c647166185bc@postgres:5432/mastodon"
# elasticsearch
ELASTIC_PASSWORD=dd0bd7a95960623ed8e084a1fb7d5c
# mastodon database configuration
#DB_HOST=pgbouncer
DB_USER=postgres
DB_NAME=mastodon
DB_PASS=0fb52834c991b5e296c647166185bc
DB_PORT=5432
REDIS_HOST=redis
REDIS_PORT=6379
CACHE_REDIS_HOST=redis-volatile
CACHE_REDIS_PORT=6379
ES_ENABLED=true
ES_HOST=elasticsearch
ES_PORT=9200
ES_USER=elastic
ES_PASS=dd0bd7a95960623ed8e084a1fb7d5c
```

Enregistrez le fichier en appuyant sur Ctrl + X et en entrant Y lorsque vous y êtes invité.

Préparer le mastodonte

Préparez les fichiers statiques à être servis par Nginx. Cette étape va prendre un certain temps car Docker extraira toutes les images pour la première fois.

```
$ docker compose run --rm shell bash -c "cp -r /opt/mastodon/public/* /static"
```

Affichez la couche de données.

```
$ docker compose up -d postgres redis redis-volatile
```

Vérifiez l'état des conteneurs.

```
$ watch docker compose ps
```

```
Every 3.0s: docker compose ps
NAME                                STATUS                                COMMAND                                SERVICE                                CREATED                                STATUS                                PORTS
mastodon-postgres-3                 Up 1m3s                               postgres                                postgres                                3m50s ago                            Up 1m3s (healthy)
mastodon-redis-3                    Up 1m3s                               redis                                   redis                                   3m50s ago                            Up 1m3s (healthy)
mastodon-redis-volatile-1           Up 1m3s                               redis-volatile                          redis-volatile                         3m50s ago                            Up 1m3s (healthy)
```

Attendez l'exécution (saine), puis appuyez sur Ctrl + C et initialisez la base de données à l'aide de la commande suivante.

```
$ docker compose run --rm shell bundle exec rake db:setup
```

Si vous obtenez l'erreur concernant la base de données `mastodonte` déjà existant, exécutez la commande suivante.

```
$ docker compose run --rm shell bundle exec rake db:migrate
```

Étape 5 - Installer Nginx

Debian 12 est livré avec une ancienne version de Nginx. Pour installer la dernière version, vous devez télécharger le référentiel officiel Nginx.

Importez la clé de signature de Nginx.

```
$ curl https://nginx.org/keys/nginx_signing.key | gpg --dearmor \
| sudo tee /usr/share/keyrings/nginx-archive-keyring.gpg >/dev/null
```

Ajoutez le référentiel pour la version principale de Nginx.

```
$ echo "deb [signed-by=/usr/share/keyrings/nginx-archive-keyring.gpg] \
http://nginx.org/packages/mainline/debian \lsb_release -cs nginx" \
| sudo tee /etc/apt/sources.list.d/nginx.list
```

Mettez à jour les référentiels système.

```
$ sudo apt update
```

Installation de Nginx

```
$ sudo apt install nginx
```

Vérifiez l'installation. Sur les systèmes Debian, vous avez besoin de sudo pour exécuter la commande suivante.

```
$ sudo nginx -v
nginx version: nginx/1.25.3
```

Démarrez le serveur Nginx.

```
$ sudo systemctl start nginx
```

Vérifiez l'état du serveur.

```
$ sudo systemctl status nginx
? nginx.service - nginx - high performance web server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; preset: enabled)
   Active: active (running) since Mon 2024-01-01 10:17:38 UTC; 4s ago
     Docs: https://nginx.org/en/docs/
   Process: 8972 ExecStart=/usr/sbin/nginx -c /etc/nginx/nginx.conf (code=exited, status=0/SUCCESS)
  Main PID: 8973 (nginx)
    Tasks: 3 (Limit: 4637)
   Memory: 2.9M
         CPU: 17ms
   CGroup: /system.slice/nginx.service
           ?78973 "nginx: master process /usr/sbin/nginx -c /etc/nginx/nginx.conf"
           ?78974 "nginx: worker process"
           ?78975 "nginx: worker process"

Jan 01 10:17:38 mastodon systemd[1]: Starting nginx.service - nginx - high performance web server...
Jan 01 10:17:38 mastodon systemd[1]: Started nginx.service - nginx - high performance web server.
```

Étape 6 - Installer SSL

Nous devons installer Certbot pour générer le certificat SSL. Vous pouvez soit installer Certbot à l'aide du référentiel Debian, soit récupérer la dernière version à l'aide de l'outil Snapd. Nous utiliserons le Snapd version.

Debian 12 n'est pas fourni avec Snapd installé. Installez le package Snapd.

```
$ sudo apt install snapd
```

Exécutez les commandes suivantes pour vous assurer que votre version de Snapd est à jour. Assurez-vous que votre version de Snapd est à jour.

```
$ sudo snap install core
$ sudo snap refresh core
```

Installez Certbot.

```
$ sudo snap install --classic certbot
```

Utilisez la commande suivante pour vous assurer que la commande Certbot s'exécute en créant un lien symbolique vers le répertoire /usr/bin.

```
$ sudo ln -s /snap/bin/certbot /usr/bin/certbot
```

Vérifiez l'installation.

```
$ certbot --version
certbot 2.8.0
```

Exécutez la commande suivante pour générer un certificat SSL.

```
$ sudo certbot certonly --nginx --agree-tos --no-eff-email --staple-ocsp --preferred-challenges http -m name@example.com -d mastodon.example.com
```

La commande ci-dessus téléchargera un certificat dans le répertoire /etc/letsencrypt/live/mastodon.example.com sur votre serveur.

Générez un certificat de groupe Diffie-Hellman.

```
$ sudo openssl dhparam -dsaparam -out /etc/ssl/certs/dhparam.pem 4096
```

Vérifiez le service de planification de renouvellement Certbot.

```
$ systemctl list-timers
```

Tu trouveras snap.certbot.renew.service comme l'un des services dont l'exécution est planifiée.

NEXT	LEFT	LAST	PASSED	UNIT	ACTIVATES
Mon 2024-01-01 20:03:52 UTC 9h left	Mon 2023-12-11 21:56:24 UTC 2 weeks 6 days ago	apt-daily.timer	apt-daily.service		
Mon 2024-01-01 21:06:00 UTC 10h left	-	snap.certbot.renew.timer	snap.certbot.renew.service		
Tue 2024-01-02 00:00:00 UTC 13h left	-	dpkg-db-backup.timer	dpkg-db-backup.service		

Effectuez un essai à sec du processus pour vérifier si le renouvellement SSL fonctionne correctement.

```
$ sudo certbot renew --dry-run
```

Si vous ne voyez aucune erreur, vous êtes prêt. Votre certificat se renouvellera automatiquement.

Étape 7 - Configurer Nginx

Ouvrir le fichier /etc/nginx/nginx.conf pour l'éditer.

```
$ sudo nano /etc/nginx/nginx.conf
```

Ajoutez la ligne suivante avant la ligne include /etc/nginx/conf.d/*.conf ;

```
server_names_hash_bucket_size 64;
```

Enregistrez le fichier en appuyant sur Ctrl + X et en entrant Y lorsque vous y êtes invité.

Créez et ouvrez le fichier /etc/nginx/conf.d/mastodon.conf pour le modifier.

```
$ sudo nano /etc/nginx/conf.d/mastodon.conf
```

Collez-y le code suivant.

```
map $http_upgrade $connection_upgrade {
    default upgrade;
    "" close;
}

upstream backend {
```

```

server 127.0.0.1:3000 fail_timeout=0;
}

upstream streaming {
server 127.0.0.1:4000 fail_timeout=0;
}

proxy_cache_path /var/cache/nginx levels=1:2 keys_zone=CACHE:10m inactive=7d max_size=1g;

server {
listen 80 default_server;
server_name mastodon.example.com;
location / { return 301 https://$host$request_uri; }
}

server {
listen 443 ssl;
server_name mastodon.example.com;

access_log /var/log/nginx/mastodon.access.log;
error_log /var/log/nginx/mastodon.error.log;

http2 on; # Enable HTTP/2 - works only on Nginx 1.25.1+

ssl_certificate /etc/letsencrypt/live/mastodon.example.com/fullchain.pem;
ssl_certificate_key /etc/letsencrypt/live/mastodon.example.com/privkey.pem;
ssl_trusted_certificate /etc/letsencrypt/live/mastodon.example.com/chain.pem;
ssl_session_timeout 1d;

# Enable TLS versions (TLSv1.3 is required upcoming HTTP/3 QUIC).
ssl_protocols TLSv1.2 TLSv1.3;

# Enable TLSv1.3's 0-RTT. Use $ssl_early_data when reverse proxying to
# prevent replay attacks.
#
# @see: https://nginx.org/en/docs/http/nginx_http_ssl_module.html#ssl_early_data
ssl_early_data on;

ssl_ciphers 'ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384';
ssl_prefer_server_ciphers on;
ssl_session_cache shared:SSL:10m;
ssl_session_tickets off;

keepalive_timeout 70;
sendfile on;
client_max_body_size 80m;

# OCSP Stapling ---
# fetch OCSP records from URL in ssl_certificate and cache them
ssl_stapling on;
ssl_stapling_verify on;
ssl_dhparam /etc/ssl/certs/dhparam.pem;
add_header X-Early-Data $tls1_3_early_data;

root /opt/mastodon/web/public;

gzip on;
gzip_disable "msie6";
gzip_vary on;
gzip_proxied any;
gzip_comp_level 6;
gzip_buffers 16 8k;
gzip_http_version 1.1;
gzip_types text/plain text/css application/json application/javascript text/xml application/xml application/xml+rss text/javascript image/svg+xml image/x-icon;
add_header Strict-Transport-Security "max-age=31536000" always;

location / {
try_files $uri @proxy;
}

location ~ ^/(system/accounts/avatars/system/media_attachments/files) {
add_header Cache-Control "public, max-age=31536000, immutable";
add_header Strict-Transport-Security "max-age=31536000" always;
root /opt/mastodon;
try_files $uri @proxy;
}

location ~ ^/(emoji/packs) {
add_header Cache-Control "public, max-age=31536000, immutable";
add_header Strict-Transport-Security "max-age=31536000" always;
try_files $uri @proxy;
}

location /sw.js {
add_header Cache-Control "public, max-age=0";
add_header Strict-Transport-Security "max-age=31536000" always;
try_files $uri @proxy;
}

location @proxy {
proxy_set_header Host $host;
proxy_set_header X-Real-IP $remote_addr;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
proxy_set_header X-Forwarded-Proto $scheme;
proxy_set_header Proxy "";
proxy_pass_header Server;

proxy_pass http://backend;
proxy_buffering on;
proxy_redirect off;
proxy_http_version 1.1;
proxy_set_header Upgrade $http_upgrade;
proxy_set_header Connection $connection_upgrade;

proxy_cache CACHE;
proxy_cache_valid 200 7d;
proxy_cache_valid 410 24h;
proxy_cache_use_stale error timeout updating http_500 http_502 http_503 http_504;
add_header X-Cached-Status $upstream_cache_status;
add_header Strict-Transport-Security "max-age=31536000" always;

tcp_nodelay on;
}

location /api/v1/streaming {
proxy_set_header Host $host;
proxy_set_header X-Real-IP $remote_addr;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
proxy_set_header X-Forwarded-Proto $scheme;
proxy_set_header Proxy "";

proxy_pass http://streaming;
proxy_buffering off;
proxy_redirect off;
proxy_http_version 1.1;
proxy_set_header Upgrade $http_upgrade;
proxy_set_header Connection $connection_upgrade;

tcp_nodelay on;
}

error_page 500 501 502 503 504 /500.html;
}

# This block is useful for debugging TLS v1.3. Please feel free to remove this
# and use the '$ssl_early_data' variable exposed by NGINX directly should you
# wish to do so.
map $ssl_early_data $tls1_3_early_data {
default "";
}

```

Une fois terminé, enregistrez le fichier en appuyant sur Ctrl + X et en entrant Y lorsque vous y êtes invité.

Vérifiez la syntaxe du fichier de configuration Nginx.

```

$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful

```

Redémarrez le serveur Nginx.

```

$ sudo systemctl restart nginx

```

Étape 8 - Démarrer Mastodon

Outil CLI Tootctl

L'outil Tootctl CLI est utilisé pour effectuer des tâches administratives sur Mastodon. Nous devons le rendre accessible sur le shell hôte.

Créez le fichier /usr/local/bin/tootctl et ouvrez-le pour le modifier.


```
$ sudo nano /usr/local/bin/tootctl
```

Collez-y le code suivant.

```
#!/bin/bash
docker compose -f /opt/mastodon/docker-compose.yml run --rm shell tootctl "$@"
```

Enregistrez le fichier en appuyant sur Ctrl + X et en entrant Y lorsque vous y êtes invité.

Donnez au fichier l'autorisation d'exécutable.

```
$ sudo chmod +x /usr/local/bin/tootctl
```

Fichier de service Mastodonte

Vous pouvez démarrer les conteneurs Mastodon à l'aide de la commande Docker compose mais c'est plus facile à faire via un fichier unité systemd.

Créez et ouvrez le fichier de service Mastodon pour le modifier.

```
$ sudo nano /etc/systemd/system/mastodon.service
```

Collez-y le code suivant.

```
[Unit]
Description=Mastodon service
After=docker.service

[Service]
Type=oneshot
RemainAfterExit=yes

WorkingDirectory=/opt/mastodon
ExecStart=/usr/bin/docker compose -f /opt/mastodon/docker-compose.yml up -d
ExecStop=/usr/bin/docker compose -f /opt/mastodon/docker-compose.yml down

[Install]
WantedBy=multi-user.target
```

Enregistrez le fichier en appuyant sur Ctrl + X et en entrant Y lorsque vous y êtes invité.

Rechargez le démon système pour lancer le fichier de service.

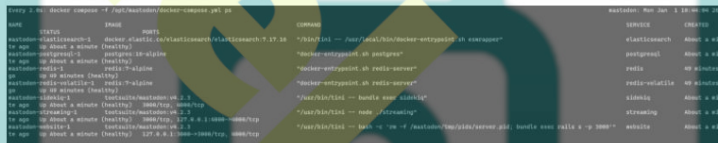
```
$ sudo systemctl daemon-reload
```

Activez et démarrez le service Mastodon.

```
$ sudo systemctl enable --now mastodon.service
```

Vérifiez l'état des conteneurs Docker.

```
$ watch docker compose -f /opt/mastodon/docker-compose.yml ps
```



Une fois que l'état des conteneurs passe à en cours d'exécution (sain), quittez l'écran en appuyant sur Ctrl + C. Créez l'utilisateur

administrateur pour Mastodon et notez le mot de passe fourni.

```
$ tootctl accounts create navjot --email name@example.com --confirmed --role Owner
OK
New password: 1338afbe1b4e06e823b6625da80cb537
```

Si vous souhaitez fermer les inscriptions des utilisateurs, utilisez la commande suivante.

```
$ tootctl settings registrations close
```

Pour rouvrir les inscriptions, exécutez la commande suivante.

```
$ tootctl settings registrations open
```

Initialiser la recherche

Vous devez créer un toot avant de pouvoir créer et remplir des index Elasticsearch. Une fois que vous avez créé un toot, exécutez la commande suivante.

```
$ tootctl search deploy
```

Vous pouvez obtenir l'erreur suivante.

```
/opt/mastodon/vendor/bundle/ruby/3.0.0/gems/ruby-progressbar-1.11.0/lib/ruby-progressbar/progress.rb:76:in `total': You can't set the item's total value to less than the current progress. (ProgressBar::InvalidProgressError)
from /opt/mastodon/vendor/bundle/ruby/3.0.0/gems/ruby-progressbar-1.11.0/lib/ruby-progressbar/base.rb:178:in `block in update_progress'
from /opt/mastodon/vendor/bundle/ruby/3.0.0/gems/ruby-progressbar-1.11.0/lib/ruby-progressbar/output.rb:43:in `with_refresh'
from /opt/mastodon/vendor/bundle/ruby/3.0.0/gems/ruby-progressbar-1.11.0/lib/ruby-progressbar/base.rb:177:in `update_progress'
from /opt/mastodon/vendor/bundle/ruby/3.0.0/gems/ruby-progressbar-1.11.0/lib/ruby-progressbar/base.rb:101:in `total='
from /opt/mastodon/lib/mastodon/search_cli.rb:67:in `deploy'
from /opt/mastodon/vendor/bundle/ruby/3.0.0/gems/thor-1.2.1/lib/thor/command.rb:27:in `run'
from /opt/mastodon/vendor/bundle/ruby/3.0.0/gems/thor-1.2.1/lib/thor/invocation.rb:127:in `invoke_command'
from /opt/mastodon/vendor/bundle/ruby/3.0.0/gems/thor-1.2.1/lib/thor.rb:392:in `dispatch'
from /opt/mastodon/vendor/bundle/ruby/3.0.0/gems/thor-1.2.1/lib/thor/invocation.rb:116:in `invoke'
from /opt/mastodon/vendor/bundle/ruby/3.0.0/gems/thor-1.2.1/lib/thor.rb:243:in `block in subcommand'
from /opt/mastodon/vendor/bundle/ruby/3.0.0/gems/thor-1.2.1/lib/thor/command.rb:27:in `run'
from /opt/mastodon/vendor/bundle/ruby/3.0.0/gems/thor-1.2.1/lib/thor/invocation.rb:127:in `invoke_command'
from /opt/mastodon/vendor/bundle/ruby/3.0.0/gems/thor-1.2.1/lib/thor.rb:392:in `dispatch'
from /opt/mastodon/vendor/bundle/ruby/3.0.0/gems/thor-1.2.1/lib/thor/base.rb:485:in `start'
from /opt/mastodon/bin/tootctl:8:in `'
```

Dans ce cas, entrez le shell du conteneur du site Web.

```
$ docker exec -it mastodon-website-1 /bin/bash
```

Exécutez la commande suivante.

```
$ sed -E '/progress.total = /d' -i lib/mastodon/search_cli.rb
```

Quittez le shell du conteneur.

```
$ exit
```

Exécutez à nouveau la commande de déploiement Elasticsearch. Parfois, la commande peut fonctionner plus tard. C'est un [problème récurrent chez Mastodon](#), il n'y a donc pas de solution définitive pour le moment.

```
$ tootctl search deploy
```

```
Done! 1/??? [-----] ETA: ??:??:?? (0 docs/s)
```

Services d'assistance supplémentaires

Créons un autre service pour supprimer les fichiers multimédias téléchargés.

Créez et ouvrez le service de suppression de médias Mastodon pour le modifier.

```
$ sudo nano /etc/systemd/system/mastodon-media-remove.service
```

Collez-y le code suivant.

```
[Unit]
Description=Mastodon - media remove service
Wants=mastodon-media-remove.timer

[Service]
Type=oneshot
StandardError=null
StandardOutput=null

WorkingDirectory=/opt/mastodon
ExecStart=/usr/bin/docker compose -f /opt/mastodon/docker-compose.yml run --rm shell tootctl media remove

[Install]
WantedBy=multi-user.target
```

Enregistrez le fichier en appuyant sur Ctrl + X et en entrant Y lorsque vous y êtes invité.

Si vous souhaitez planifier la suppression du média, vous pouvez configurer un service de minuterie pour cela.

```
$ sudo nano /etc/systemd/system/mastodon-media-remove.timer
```

Collez le code suivant.

```
[Unit]
Description=Schedule a media remove every week

[Timer]
Persistent=true
OnCalendar=Sat *.*.* 00:00:00
Unit=mastodon-media-remove.service

[Install]
WantedBy=timers.target
```

Enregistrez le fichier en appuyant sur Ctrl + X et en entrant Y lorsque vous y êtes invité.

Vous pouvez configurer un autre service pour supprimer les cartes d'aperçu enrichi générées à l'aide des balises OpenGraph.

```
$ sudo nano /etc/systemd/system/mastodon-preview_cards-remove.service
```

Collez le code suivant.

```
[Unit]
Description=Mastodon - preview cards remove service
Wants=mastodon-preview_cards-remove.timer

[Service]
Type=oneshot
StandardError=null
StandardOutput=null

WorkingDirectory=/opt/mastodon
ExecStart=/usr/bin/docker compose -f /opt/mastodon/docker-compose.yml run --rm shell tootctl preview_cards remove

[Install]
WantedBy=multi-user.target
```

Enregistrez le fichier en appuyant sur Ctrl + X et en entrant Y lorsque vous y êtes invité.

Réglez le service de minuterie correspondant.

```
$ sudo nano /etc/systemd/system/mastodon-preview_cards-remove.timer
```

Collez le code suivant.

```
[Unit]
Description=Schedule a preview cards remove every week

[Timer]
Persistent=true
OnCalendar=Sat *.*.* 00:00:00
Unit=mastodon-preview_cards-remove.service

[Install]
WantedBy=timers.target
```

Enregistrez le fichier en appuyant sur Ctrl + X et en entrant Y lorsque vous y êtes invité.

Rechargez le démon système.

```
$ sudo systemctl daemon-reload
```

Activez et démarrez les minuteries.

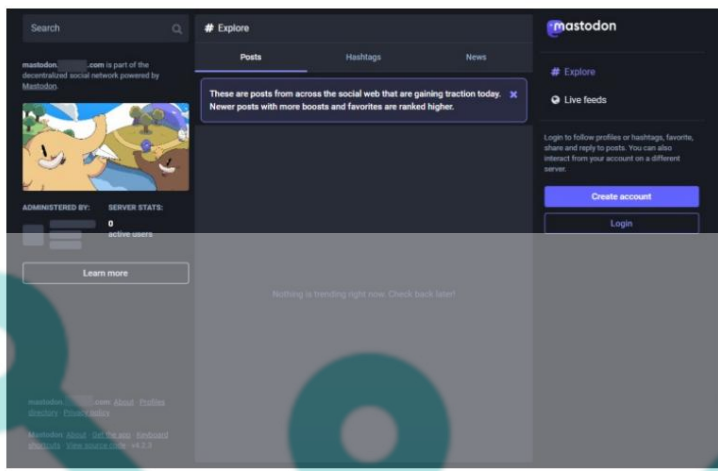
```
$ sudo systemctl enable --now mastodon-preview_cards-remove.timer
$ sudo systemctl enable --now mastodon-media-remove.timer
```

Listez tous les minuteurs pour vérifier l'horaire des services Mastodon.

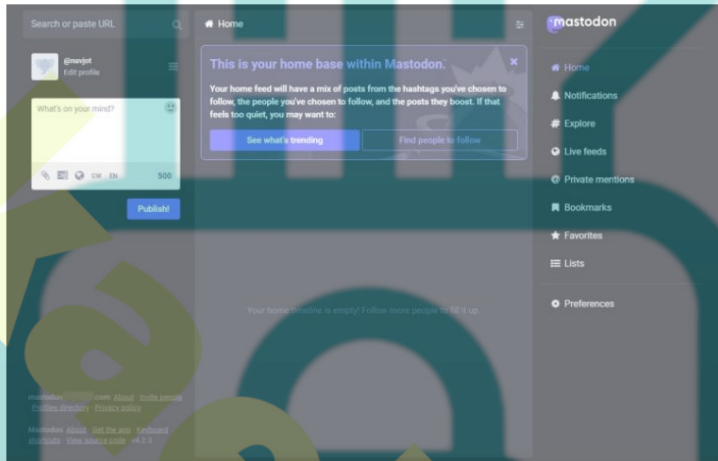
```
$ systemctl list-timers
.....
Sat 2024-01-06 00:00:00 UTC 4 days left - - mastodon-media-remove.timer mastodon-media-remove.service
Sat 2024-01-06 00:00:00 UTC 4 days left - - mastodon-preview_cards-remove.timer mastodon-preview_cards-remove.service
```

Accéder à Mastodonte

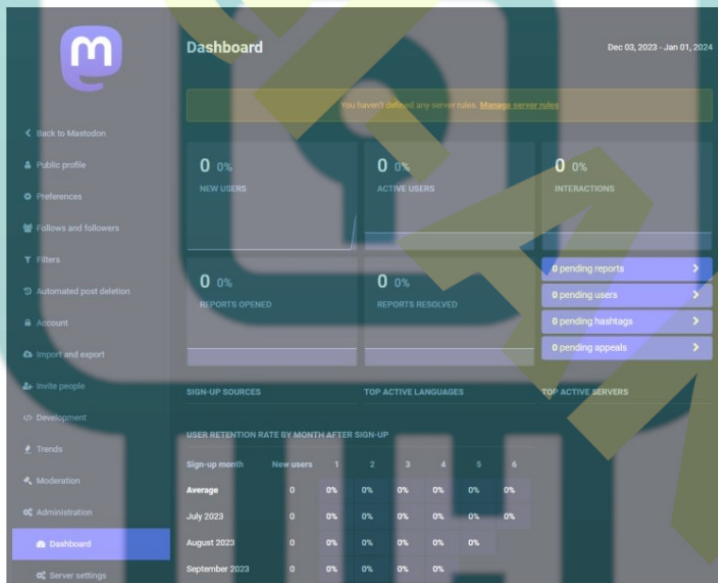
Visitez l'URL <https://mastodon.example.com> pour accéder à votre instance et vous verrez une page similaire.



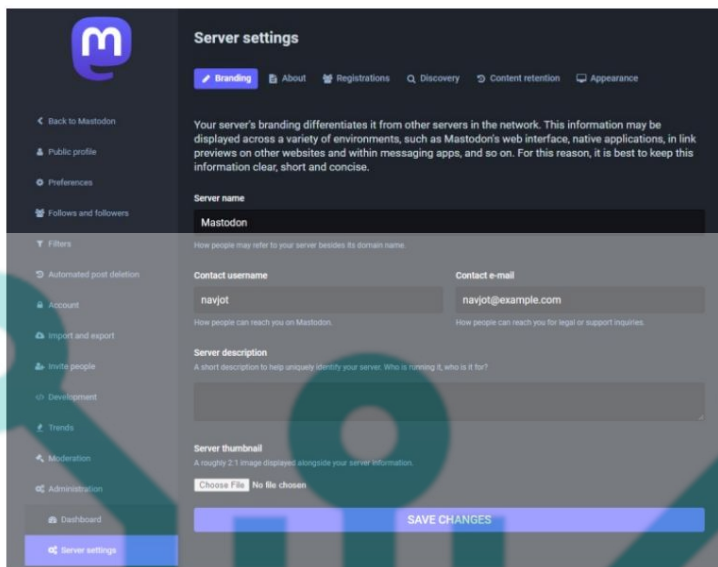
Dans la capture d'écran ci-dessus, vous pouvez voir qu'il y a 0 utilisateur. C'est parce que nous ne sommes pas encore connectés. Même si vous créez un compte administrateur, il n'apparaît pas sur la page principale lors de la première exécution. Pour ce faire, connectez-vous à votre instance et vous serez redirigé vers la page suivante.



Cliquez sur l'option Préférences dans la barre latérale droite pour accéder aux paramètres. De là, cliquez sur l'option Administration dans le menu de gauche pour accéder au panneau d'administration de Mastodon.



Cliquez sur l'option Paramètres du serveur dans la barre latérale gauche.

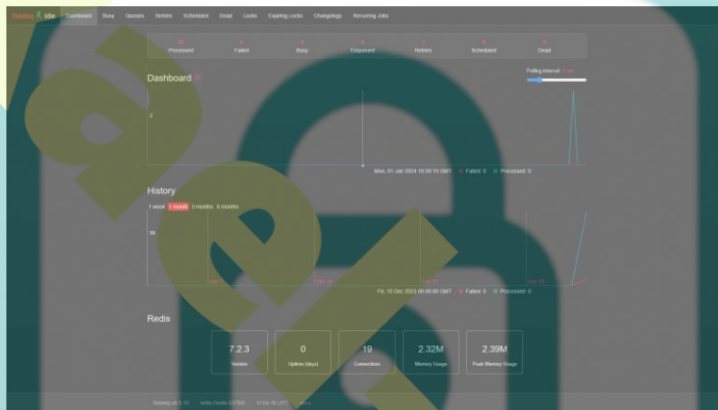


Ici, remplissez votre nom d'utilisateur de contact et votre adresse e-mail professionnelle qui seront désormais reflétés sur la page d'accueil de votre serveur. Remplissez également diverses autres informations, notamment la description du serveur, le logo et règles de serveur pour personnaliser votre instance Mastodon.

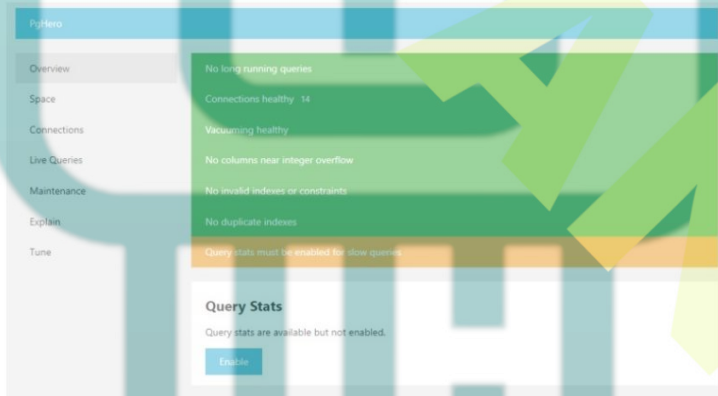
Étape 9 - Entretien du mastodonte

Pour afficher les performances et les journaux de votre instance Mastodon, rendez-vous sur <https://mastodon.example.com/sidekiq/>

Ici, vous pouvez afficher une liste de divers processus et tâches planifiées liés à votre instance Mastodon. Vous pouvez également rechercher les tâches ayant échoué dans la section Mortes ou Nouvelles tentatives . Il vous dira également l'utilisation de la mémoire de votre instance.



Vous pouvez vérifier la santé de la base de données de votre instance à partir de <https://mastodon.example.com/pghero/>



Vous pouvez effectuer la maintenance de votre base de données, exécuter des requêtes SQL et supprimer les index inutilisés. Pour activer les statistiques de requête, cliquez sur le bouton Activer sur la page ci-dessus et vous obtiendrez le information suivante.

Query Stats

Make them available by adding the following lines to `postgresql.conf`:

```
shared_preload_libraries = 'pg_stat_statements'
pg_stat_statements.track = all
```

Restart the server for the changes to take effect.

Passez à l'utilisateur root.

```
$ sudo -i su
```

Basculez vers le répertoire `/opt/mastodon/database/postgresql`.

```
$ cd /opt/mastodon/database/postgresql
```

Ouvrez le fichier `postgresql.conf`.

```
$ nano postgresql.conf
```

Recherchez la ligne `#shared_preload_libraries = ''` (la modification nécessite un redémarrage) et remplacez-la par ce qui suit.

```
shared_preload_libraries = 'pg_stat_statements'
```

Ajoutez la ligne suivante à la fin du fichier.

```
pg_stat_statements.track = all
```

Enregistrez le fichier en appuyant sur Ctrl + X et en entrant Y lorsque vous y êtes invité.

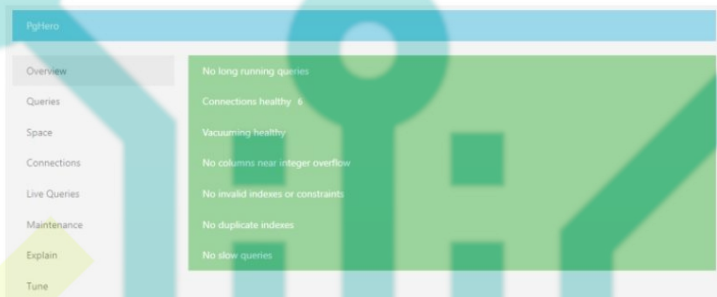
Redémarrez les conteneurs Mastodon.

```
$ systemctl restart mastodon.service
```

Quittez le shell racine.

```
$ exit
```

Si vous consultez la page d'intégrité de la base de données, vous pouvez voir s'il y a maintenant des requêtes lentes.



Remarque : Vous pouvez également lancer les URL PgHero et Sidekiq depuis le menu Préférences .

Si votre site ne se charge pas pour une raison quelconque, vous pouvez consulter les journaux générés par Docker.

```
$ docker registre <nom du conteneur>
```

Étape 10 - Sauvegarder Mastodon

Nous utiliserons un outil tiers appelé [Restic](#) pour sauvegarder Mastodon. La première étape pour sauvegarder à l'aide de Restic consiste à ajouter tous les fichiers et répertoires à la liste des référentiels.

Créez et ouvrez le fichier de liste de référentiels pour le modifier.

```
$ sudo nano /opt/mastodon/backup-files
```

Collez-y les lignes suivantes.

```
/etc/nginx
/etc/letsencrypt
/etc/systemd/system
/root
/opt/mastodon/database/pgbackups
/opt/mastodon/*.env
/opt/mastodon/docker-compose.yml
/opt/mastodon/branding
/opt/mastodon/database/redis
/opt/mastodon/web/system
/opt/mastodon/backup-files
/opt/mastodon/mastodon-backup
```

Enregistrez le fichier en appuyant sur Ctrl + X et en entrant Y lorsque vous y êtes invité.

Installez Restic.

```
$ sudo apt install restic
```

Créez un référentiel de sauvegarde et créez la sauvegarde initiale. Nous sauvegardons nos données sur le service S3.

```
$ restic -r s3:https://$SERVER:$PORT/mybucket init
$ restic -r s3:https://$SERVER:$PORT/mybucket backup $(cat /opt/mastodon/backup-files) --exclude /opt/mastodon/database/postgresql
```

Créez un minuteur de service de sauvegarde Mastodon et ouvrez-le pour le modifier.

```
$ sudo nano /etc/systemd/system/mastodon-backup.timer
```

Collez-y le code suivant.

```
[Unit]
Description=Schedule a mastodon backup every hour

[Timer]
Persistent=true
OnCalendar=*:00:00
Unit=mastodon-backup.service

[Install]
WantedBy=timers.target
```

Enregistrez le fichier en appuyant sur Ctrl + X et en entrant Y lorsque vous y êtes invité.

Créez un fichier de service de sauvegarde Mastodon et ouvrez-le pour le modifier.

```
$ sudo nano /etc/systemd/system/mastodon-backup.service
```

Collez-y le code suivant.

```
[Unit]
Description=Mastodon - backup service
# Without this, they can run at the same time and race to docker compose,
# double-creating networks and failing due to ambiguous network definition
# requiring `docker network prune` and restarting
After=mastodon.service

[Service]
Type=oneshot
StandardErrorFile=/var/log/mastodon-backup.err
StandardOutputFile=/var/log/mastodon-backup.log

WorkingDirectory=/opt/mastodon
ExecStart=/bin/bash /opt/mastodon/mastodon-backup

[Install]
WantedBy=multi-user.target
```

Enregistrez le fichier en appuyant sur Ctrl + X et en entrant Y lorsque vous y êtes invité.

Ensuite, créez et ouvrez le fichier `/opt/mastodon/mastodon-backup` pour le modifier. Celui-ci contient les commandes de sauvegarde réelles.


```
$ sudo nano /opt/mastodon/mastodon-backup
```

Collez-y le code suivant.

```
#!/bin/bash
set -e
AWS_ACCESS_KEY_ID=
AWS_SECRET_ACCESS_KEY=
SERVER=
PORT=
RESTIC_PASSWORD_FILE=/root/restic-password

docker compose -f /opt/mastodon/docker-compose.yml run --rm postgresql sh -c "pg_dump -Fp mastodon | gzip > /backups/dump.sql.gz"
restic -r s3:https://$SERVER:$PORT/mybucket --cache-dir=/root backup $(cat /opt/mastodon/backup-files) --exclude /opt/mastodon/database/postgresql
restic -r s3:https://$SERVER:$PORT/mybucket --cache-dir=/root forget --prune --keep-hourly 24 --keep-daily 7 --keep-monthly 3
```

Enregistrez le fichier en appuyant sur Ctrl + X et en entrant Y lorsque vous y êtes invité.

Donnez des autorisations exécutables au script de sauvegarde.

```
$ sudo chmod +x /opt/mastodon/mastodon-backup
```

Rechargez le service daemon et démarrez le service de sauvegarde et le minuteur.

```
$ sudo systemctl daemon-reload
$ sudo systemctl enable --now mastodon-backup.service
$ sudo systemctl enable --now mastodon-backup.timer
```

Confirmez que des sauvegardes horaires sont en cours et accessibles à l'aide des commandes suivantes.

```
$ restic -r s3:https://$SERVER:$PORT/mybucket snapshots
$ restic -r s3:https://$SERVER:$PORT/mybucket mount /mnt
```

Étape 11 - Améliorer Mastodon

La mise à niveau de Mastodon nécessite plusieurs étapes. Tout d'abord, passez au répertoire.

```
$ cd /opt/mastodon
```

Extrayez les dernières images de conteneur pour Mastodon.

```
$ docker compose pull
```

Apportez des modifications au docker-compose.yml si vous le souhaitez.

Effectuez toutes les migrations de bases de données.

```
$ docker compose run --rm shell bundle exec rake db:migrate
```

Mettez à jour vos copies de fichiers statiques.

```
$ docker compose run --rm shell bundle exec rake db:migrate
```

Redémarrez les conteneurs Mastodon.

```
$ sudo systemctl restart mastodon.service
```

Les instructions ci-dessus sont des instructions de mise à jour génériques. Vérifiez toujours la page [des versions GitHub de Mastodon](#) pour rechercher des tâches et des commandes de mise à jour spécifiques entre les versions pour garantir tout se passe bien.

Conclusion

Ceci conclut notre tutoriel sur l'installation de Mastodon Social Network à l'aide de Docker sur un serveur Debian 12. Si vous avez des questions, postez-les dans les commentaires ci-dessous.