Tutoriels

- Mise en place d'un firewall avec UFW
- Installation de Docker et Docker-Compose sur Ubuntu
- Génération d'un certificat auto signé
- Installation de Let's Encrypt / Certbot et génération d'un certificat
- Déplacer les overlays Docker dans un autre montage
- Installer un relais backup MX Postfix
- Installation d'une stack monitoring Netdata/Prometheus/Grafana
- Installer ZSH et Oh My ZSH
- Mise en place d'Authelia : OpenSource SSO
- Initialisation Serveur Ubuntu
- Formation Linux
- Synchroniser la librairie Netfloux avec KyBook sur iOS
- Init Serveur Ubuntu pour Web Hosting

Mise en place d'un firewall avec UFW

UFW est un outil en ligne de commande, alternative a iptables, permettant de mettre en place rapidement des règles de flux basiques.

Installation d'UFW

```
apt install ufw
```

Configuration initiale

Nous allons configurer le firewall pour que le comportement par défaut bloque tous les flux entrants et ouvre les flux sortant :

```
ufw default allow outgoing && ufw default deny incoming
```

Installation de Docker et Docker-Compose sur Ubuntu

Installation de Docker

Suppression et nettoyage des anciennes releases

```
apt-get remove docker docker-engine docker.io containerd runc
```

Installation des dépendances

```
apt-get install -y apt-transport-https ca-certificates curl gnupg-agent software-properties-common
```

```
# Add Docker's official GPG key:
sudo apt-get update
sudo apt-get install ca-certificates curl gnupg
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
sudo chmod a+r /etc/apt/keyrings/docker.gpg

# Add the repository to Apt sources:
echo \
  "deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/ubuntu \
  "$(. /etc/os-release && echo "$VERSION_CODENAME")" stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
```

Installation de docker

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

Installation de Docker-compose

Avant de copier / coller ces lignes, vérifiez si la version est bien la dernière disponible sur la pages des releases.

> A ce jour, nous sommes en v2.23.3

Récupération du binaire de docker-compose

```
curl -s https://api.github.com/repos/docker/compose/releases/latest | grep "linux-$(uname -m)" | grep -v "name" | head -1 | grep "browser_download_url" | awk '{print $2}' | sed 's/\"//g' | wget -qi - -O ~/.local/bin/docker-compose && chmod +x ~/.local/bin/docker-compose && ~/.local/bin/docker-compose --version
```

A ce stade, vous devriez être ready to contenerize :

```
❯ docker --version
Docker version 20.10.21, build baeda1f
❯ docker-compose --version
Docker Compose version v2.13.0
```

Utilisation de Docker & Docker-compose sans sudo

Créer le groupe docker si il n'existe pas déjà :

```
sudo groupadd docker
```

Ajouter votre utilisateur au groupe docker :

```
sudo usermod -aG docker monuser
```

Pour appliquer les modifications, rechargez votre shell ou déconnectez-vous de votre session. Vous devriez être en mesure de lancer docker sans sudo.

Génération d'un certificat auto signé

Prérequis

```
apt install openssl
```

Génération de la clé privée

```
openssl genrsa -aes256 -out certificat.key 4096
```

Génération du fichier de demande de signature (CSR)

```
openssl req -new -key certificat.key.lock -out certificat.csr
```

Génération du certificat signé

```
openssl x509 -req -days 365 -in certificat.csr -signkey certificat.key.lock -out certificat.crt
```

Installation de Let's Encrypt / Certbot et génération d'un certificat

Installation de Certbot via les repos Officiels

Mise à jour du système et installation des prérequis

```
apt-get update
apt-get install software-properties-common
```

Installation des repos Cerbot

```
add-apt-repository ppa:certbot/certbot
apt-get update
```

Installation de certbot

```
apt-get install certbot
```

Installation de Certbot via Pyhton-pip

Installation des prérequis

```
apt-get update
apt-get install software-properties-common python3-pip
```

Installation de Certbot

```
pip3 install --upgrade certbot
```

Génération d'un certificat Let's Encrypt en challenge HTTP

Pour générer un certificat Let's Encrypt avec le challenge HTTP, assurez-vous d'avoir un Vhost écoutant sur le port 80 et que le nom de domaine désiré pointe bien sur l'IP publique du serveur. Ensuite, ajoutez une location dans la configuration Nginx pour autoriser le challenge :

```
location /.well-known/acme-challenge/ {
root /var/www/html;
}
```

Créez le répertoire :

```
mkdir -p /var/www/html/.well-known/acme-challenge/
chown www-data: /var/www/html/.well-known/acme-challenge/
chmod 0775 /var/www/html/.well-known/acme-challenge/
```

Générez le certificat :

```
certbot certonly --webroot --agree-tos --email email@example.com -w /var/www/html/example.com -d example.com
```

Génération d'un certificat Let's Encrypt avec le plugin Nginx

```
apt install python-certbot-nginx
```

Avant de pouvoir générer le certificat avec le plugin Nginx, assurez-vous d'avoir configuré un vhost Nginx avec le servername désiré, écoutant sur le port 80. Ensuite, procédez à la génération du certificat :

```
certbot --nginx -d example.com -d www.example.com
```

Génération d'un certificat Let's Encrypt avec le plugin DNS Cloudflare

Installation du paquet pip certbot-dns-cloudflare :

```
pip3 install certbot-dns-cloudflare --upgrade
```

Créez un fichier dans /root/.cloudflare.creds avec le contenu suivant :

```
dns_cloudflare_email = youremail@example.com
dns_cloudflare_api_key = ****************************************
```

Modifiez les droits sur le fichier :

```
chmod 0400 /root/.cloudflare.creds
```

Générez le certificat en spécifiant le challenge à utiliser :

```
certbot certonly --dns-cloudflare --dns-cloudflare-credentials /root/.cloudflare.creds -d example.com -d www.example.com
```

Pour générer un certificat Wildcard en challenge DNS :

```
certbot certonly --dns-cloudflare --dns-cloudflare-credentials /root/.cloudflare.creds -d example.com -d *.example.com
```

Déplacer les overlays Docker dans un autre montage

Arrêter le service docker :

```
systemctl stop docker.socker
```

Editer le fichier **/etc/docker/daemon.json** et y ajouter :

```
{
  "data-root": "/data/docker",
}
```

Synchroniser le /var/lib/docker dans le nouveau répertoire :

```
rsync -aP /var/lib/docker/ /data/docker/
```

Redémarrer docker :

```
systemctl restart docker.service
```

Installer un relais backup MX Postfix

Prérequis

- Un nom de domaine
- Un serveur MX principal
- Ouverture du port 25 sur le MX backup
- Un enregistrement MX sur le Postfix principal

```
host autographe.com
autographe.com has address 104.21.87.75
autographe.com has address 172.67.142.90
autographe.com has IPv6 address 2606:4700:3036::6815:574b
autographe.com has IPv6 address 2606:4700:3031::ac43:8e5a
autographe.com mail is handled by 10 mx1.bldwebagency.fr.
```

Création du record DNS pour le MX secondaire

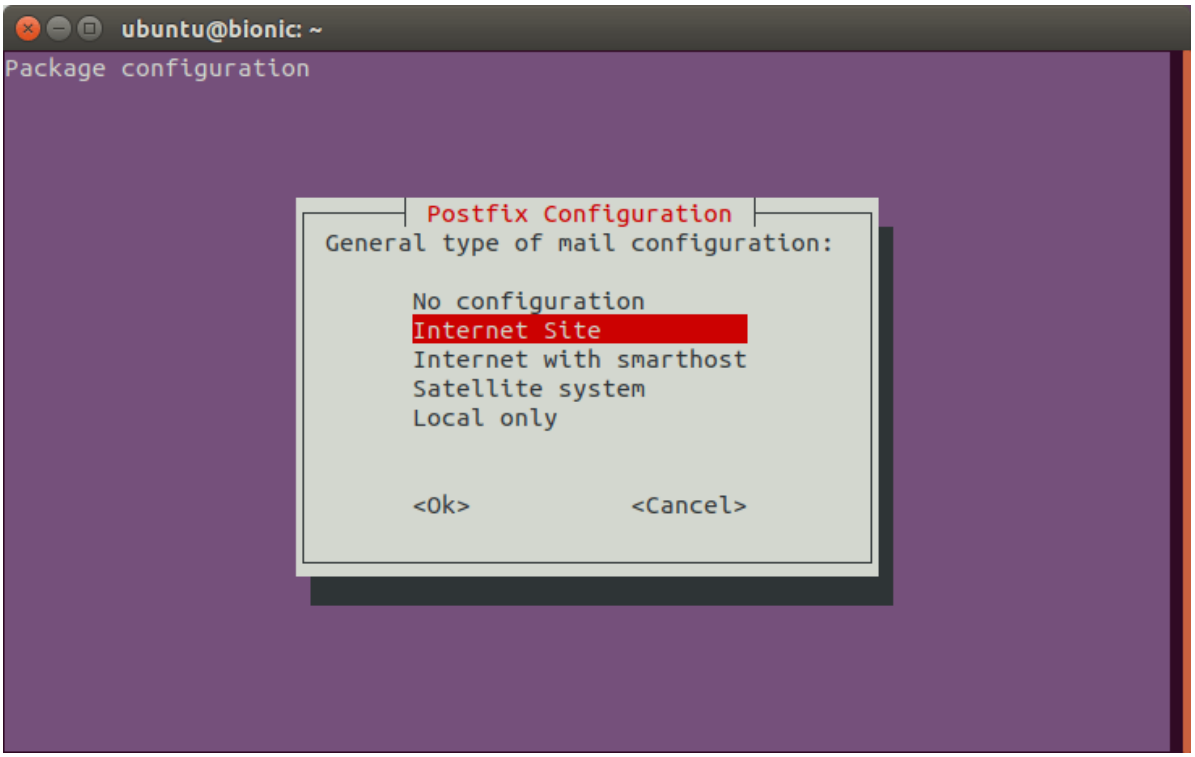
Nous allons créer dans la zone DNS un enregistrement pour le MX secondaire. Dans notre exemple nous allons donc ajouter :

```
mx2.autographe.com A 91.121.84.91
```

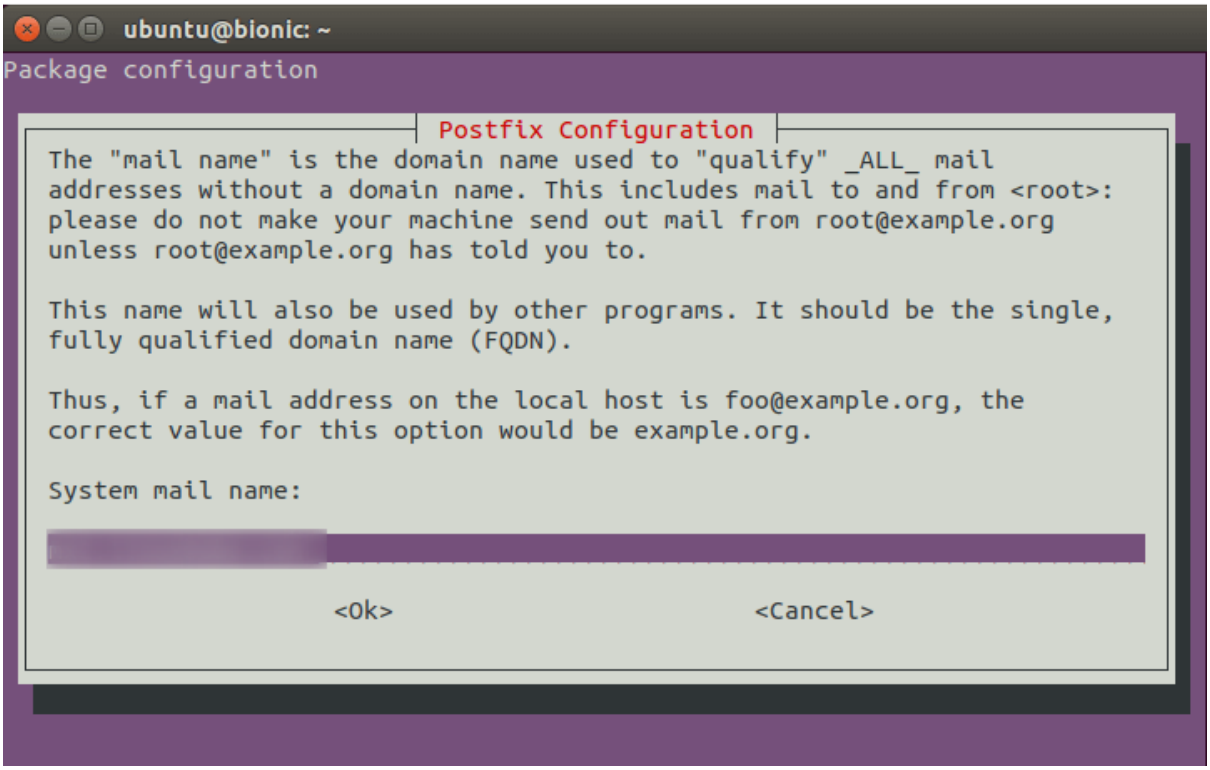
Installation de postfix sur le backup MX

```
apt install postfix
```

Dans la liste des proposition sur la configuration postfix, sélectionner **Internet Site** :



Ensuite, renseigner le nom de domaine du MX secondaire : mx2.bldwebagency.fr :



Configuration initiale de postfix

Nous allons ajouter quelques lignes à la configuration postfix (**/etc/postfix/main.cf**) par défaut pour autoriser les domaines à être relayés.

```
relay_domains = bldwebagency.fr, bldwebagency.com, autographe.com, domain3.com
myhostname = mx2.bldwebagency.fr
mydestination = $myhostname, mx2.bldwebagency.fr, localhost, localhost.localdomain, localhost
```

Configuration de la durée de vie des mails sur le backup :

```
maximal_queue_lifetime = 10d
```

Enfin, redémarrer postfix :

```
systemctl restart postfix
```

Configuration des relay recipients

Par défaut, postfix va relayer tous ce qui concerne les domaines renseignés dans relay_domains. Afin de sécuriser un peu la solution, nous allons restreindre cela aux adresses mails souhaitées. Ouvrez à nouveau le fichier **/etc/postfix/main.cf** :

```
relay_recipient_maps = hash:/etc/postfix/relay_recipients
```

Créez ensuite le fichier **/etc/postfix/relay_recipients** et ajoutez-y les adresses souhaitées, sous la forme :

```
user1@your-domain.com      OK
user2@your-domain.com      OK
user3@your-domain.com      OK
```

> Postfix prend en charge les wildards. Pour autoriser l'ensemble des adresses d'un domaine, ajoutez le domaine de cette façon :

```
user1@your-domain.com      OK
user2@your-domain.com      OK
user3@your-domain.com      OK
@2nd-domain.com            OK
```

Enfin, créez le fichier **relay_recipients.db** avec la commande suivante :

```
postmap /etc/postfix/relay_recipients
```

Activer l'encryption TLS sur le MX principal

Génération d'un certificat avec Certbot

Nous allons activer TLS sur ce backup MX. Pour cela, nous utiliserons **Certbot** pour générer gratuitement un certificat SSL :

```
apt install software-properties-common
add-apt-repository ppa:certbot/certbot
apt update
apt install certbot
```

Génération du certificat :

```
certbot certonly --standalone --preferred-challenges http --agree-tos --email contact@bldwebagency.fr -d mx2.bldwebagency.fr
```

Installation du certificat dans Postfix

Ouvrez à nouveau le fichier **/etc/postfix/main.cf** et ajoutez-y les lignes suivantes :

```
smtpd_tls_cert_file=/etc/letsencrypt/live/mx2.bldwebagency.fr/fullchain.pem
smtpd_tls_key_file=/etc/letsencrypt/live/mx2.bldwebagency.fr/privkey.pem
```

Affinez la configuration TLS avec ces lignes :

```
smtpd_tls_security_level=may
smtpd_tls_protocols = !SSLv2, !SSLv3 !TLSv1
smtpd_tls_loglevel = 1
```

Puis redémarrez Postfix :

```
systemctl restart postfix
```

Activer les échanges TLS entre le MX principal et le backup MX

Ajoutez ces lignes au fichier /etc/postfix/main.cf :

```
smtp_tls_security_level = verify
smtp_tls_verify_cert_match = hostname, nexthop, dot-nexthop
smtp_tls_CApath = /etc/ssl/certs
smtp_tls_loglevel = 1
```

Créez le lien symbolique hash :

```
openssl rehash /etc/ssl/certs
```

Redémarrez postfix :

```
systemctl restart postfix
```

Installation d'une stack monitoring Netdata/Prometheus/Grafana

```
bash <(curl -Ss https://my-netdata.io/kickstart.sh)
```

Installer ZSH et Oh My ZSH

Installation de ZSH

```
apt install zsh git curl wget
```

Installation de Oh My ZSH

```
sh -c "$(curl -fsSL https://raw.github.com/ohmyzsh/ohmyzsh/master/tools/install.sh)"
```

Modifier le thème du Shell ZSH

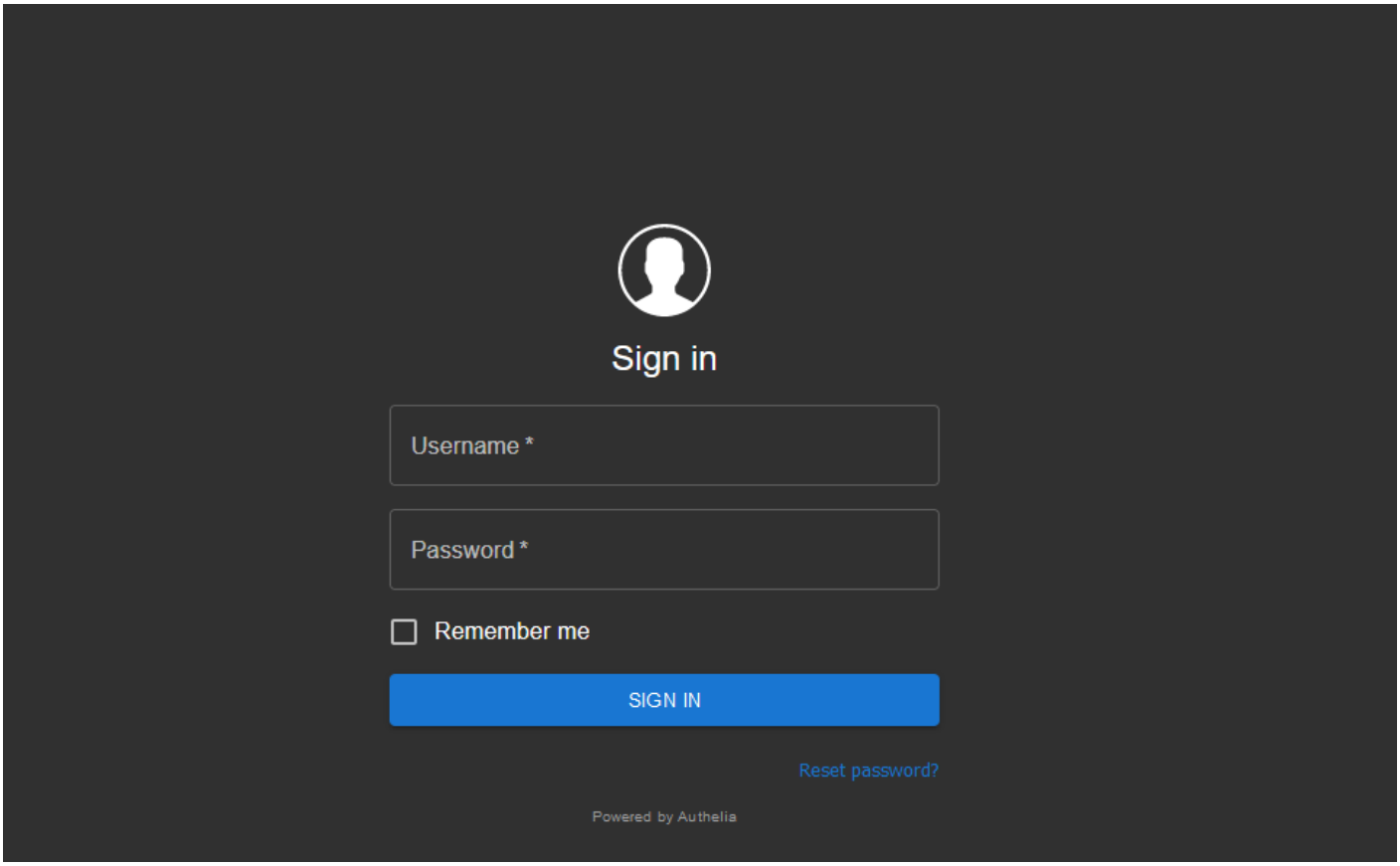
```
sed -i 's/ZSH_THEME=\"robbyrussell\"/ZSH_THEME=\"alanpeabody\"/g' ~/.zshrc && source ~/.zshrc
```

Sources

- https://ohmyz.sh/#install
- https://github.com/ohmyzsh/ohmyzsh/wiki/Themes

Mise en place d'Authelia : OpenSource SSO

Authelia est une solution OpenSource qui agit comme un portail d'accès avec authentification, ou SSO. Il permet de centraliser l'authentification des utilisateurs et leur permet l'accès à des ressources protégées. L'authentification peut passer par une simple connexion user / password mais des fonctionnalités avancées sont disponibles : authentification à deux facteurs, utilisation d'une notification push Duo ou activation d'une clé de sécurité Yubikey.



Environnement

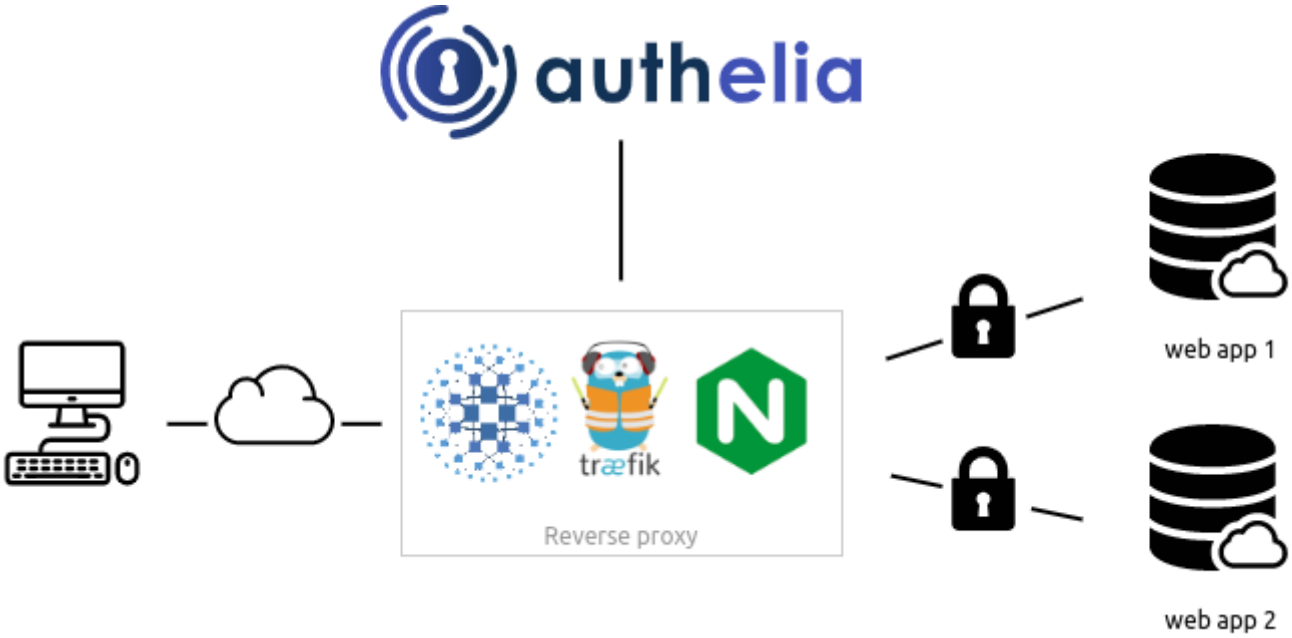
Dans cette documentation, notre architecture est la suivante, adaptez les valeurs pour "**copcol**" comme un enfant :

```
# Emplacement de la configuration des Dockers
export AUTHELIA_DOCKER_DIR="/data/Dockers/authelia"
export AUTHELIA_DOCKER_DIR_CONF="/data/Dockers/authelia/config"
```

```
export AUTHELIA_DOCKER_DIR_REDIS="/data/Dockers/authelia/redis"

# Emplacement de la stack Docker-compose
export AUTHELIA_DOCKERCOMPOSE_DIR="/opt/docker-compose"

# Domaines
export AUTHELIA_ROOT_DOMAIN="domain.com"
export AUTHELIA_AUTH_DOMAIN="auth.${AUTHELIA_ROOT_DOMAIN}"
```

Fonctionnement

Voici un exemple de mise en place d'Authelia avec Traefic / Nginx :



Mise en place d'authelia

Préparation des dossiers de configuration

Nous allons créer les dossiers de configuration et y créer les fichiers de base **configuration.yml** et **users_database.yml** :

```
mkdir -p ${AUTHELIA_DOCKER_DIR}/config
touch ${AUTHELIA_DOCKER_DIR}/config/configuration.yml
touch ${AUTHELIA_DOCKER_DIR}/config/users_database.yml
```

configuration.yaml - Configuration d'authelia

Le fichier de configuration **configuration.yml** (Pour générer les tokens suivez le guide)

```
############################################################################
############
#                Authelia configuration                            #
############################################################################
############

host: 0.0.0.0
port: 9091
log_level: info
jwt_secret: A4gYb7QFpbfKaNWAX7P7FX5y
default_redirection_url: https://auth.domain.com
totp:
  issuer: domain.com
  period: 30
  skew: 1

#duo_api:
#  hostname: api-123456789.example.com
#  integration_key: ABCDEF
#  secret_key: yet-another-long-string-of-characters-and-numbers-and-symbols

authentication_backend:
  disable_reset_password: false
  file:
    path: /config/users_database.yml
    password:
      algorithm: argon2id
      iterations: 1
      salt_length: 16
      parallelism: 8
      memory: 64

access_control:
  default_policy: deny
  rules:
    - domain:
        - "radarr.domain.com"
        - "sonarr.domain.com"
```

```
        - "radarr.domain.com"
      policy: bypass
      resources:
        - "^/api.*"
    - domain:
        - "auth.domain.com"
        - "www.domain.com"
      policy: bypass
    - domain:
        - "radarr.domain.com"
        - "sonarr.domain.com"
        - "deluge.domain.com"
      policy: one_factor
      subject:
        - ["group:admins", "group:users"]

session:
  name: authelia_session
  secret: quaeS9MaixieL1aelee0vov3J
  expiration: 3600 # 1 hour
  inactivity: 7200 # 2 hours
  domain: domain.com # Root domain

  redis:
    host: redis
    port: 6379

regulation:
  max_retries: 5
  find_time: 2m
  ban_time: 10m

theme: dark   # options: dark, light, grey

storage:
  local:
    path: /config/db.sqlite3

notifier: # Permet la validation d'un compte si 2FA
  # filesystem:
```

```
  #   filename: /config/notification.txt
  smtp:
    username: contact@domain.com
    password: Be1zah2iek7pheNgeileosaev
    host: mail.domain.com
    port: 587  # 25 non-ssl, 443 ssl, 587 tls
    sender: contact@domain.com
    subject: "[Authelia] {title}"
    disable_require_tls: false # set to true if your domain uses no tls or ssl only
    disable_html_emails: false # set to true if you don't want html in your emails
    tls:
      server_name: mail.domain.com
      skip_verify: false
      minimum_version: TLS1.2
```

users_database.yml - Base utilisateurs Authelia

Nous allons générer un fichier pour stocker les utilisateurs et groupes pour Authelia :

```
###############################################################
#                   Users Database                            #
###############################################################

# This file can be used if you do not have an LDAP set up.

# List of users
users:
  johndoe:
    displayname: "John Doe"
    password: "$argon2id$v=19$m=1048576,t=1,p=8$MFJSeXh0V2VKVWZEZFJiZg$EOSz2OgjIIV//MWf8"
    email: johndoe@domain.com
    groups:
      - admins
      - users
```

Pour générer le Hash du password, exécutez la commande suivante :

```
docker run --rm authelia/authelia:latest authelia hash-password 'votre-mot-de-passe'
```

Mise en place de la stack Docker-compose

Voici un exemple de docker-compose pour Authelia et son gestionnaire de session Redis :

```
version: '3.3'
services:
  authelia:
    container_name: authelia
    image: authelia/authelia
    restart: always
    volumes:
      - /data/Dockers/authelia/config:/config
    ports:
      - 9091:9091
    healthcheck:
      disable: true
    environment:
      - TZ=Europe/Paris
    depends_on:
      - redis
  redis:
    container_name: redis
    image: redis:alpine
    restart: always
    volumes:
      - /data/Dockers/authelia/redis:/data
    expose:
      - 6379
    environment:
      - TZ=Europe/Paris
```

Démarrez la stack :

```
docker-compose -p authelia -f ${AUTHELIA_DOCKERCOMPOSE_DIR}/authelia.yml up -d redis
docker-compose -p authelia -f ${AUTHELIA_DOCKERCOMPOSE_DIR}/authelia.yml up -d authelia
```

Configuration d'Authelia sur Nginx Proxy Manager

Pour qu'Authelia puisse être fonctionnel sur le sous-domaine / domaine choisi, il doit être listé dans la liste des access control, et une configuration Nginx doit être ajoutée.

Création de la configuration Nginx pour auth.domain.com

Créer une configuration reverse proxy pour le domaine **auth.domain.com** en upstream sur notre docker **authelia** port **9091** puis ajoutez la configuration suivante :

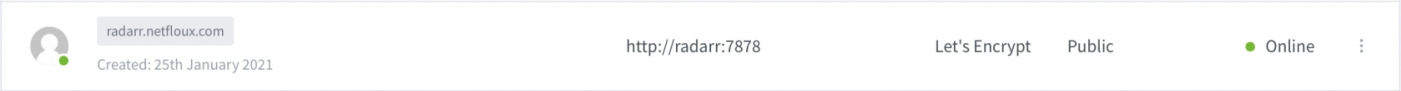
```
location / {
    set $upstream_authelia http://authelia:9091; # Adapter l'upstream authelia
    proxy_pass $upstream_authelia;
    client_body_buffer_size 128k;

    #Timeout if the real server is dead
    proxy_next_upstream error timeout invalid_header http_500 http_502 http_503;

    # Advanced Proxy Config
    send_timeout 5m;
    proxy_read_timeout 360;
    proxy_send_timeout 360;
    proxy_connect_timeout 360;

    # Basic Proxy Config
    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 X-Forwarded-Host $http_host;
    proxy_set_header X-Forwarded-Uri $request_uri;
    proxy_set_header X-Forwarded-Ssl on;
    proxy_redirect  http://  $scheme://;
    proxy_http_version 1.1;
    proxy_set_header Connection "";
    proxy_cache_bypass $cookie_session;
    proxy_no_cache $cookie_session;
    proxy_buffers 64 256k;

    # If behind reverse proxy, forwards the correct IP
    set_real_ip_from 10.0.0.0/8;
    set_real_ip_from 172.0.0.0/8;
    set_real_ip_from 192.168.0.0/16;
    set_real_ip_from fc00::/7;
    real_ip_header X-Forwarded-For;
```

```
    real_ip_recursive on;
}
```

Cette configuration permet la mise en place de l'authentification via Authelia sur Nginx pour le domaine d'auth.

Sous domaine : radarr.domain.com

Admettons que votre Proxy Host est déjà configuré sur Nginx Proxy Manager :



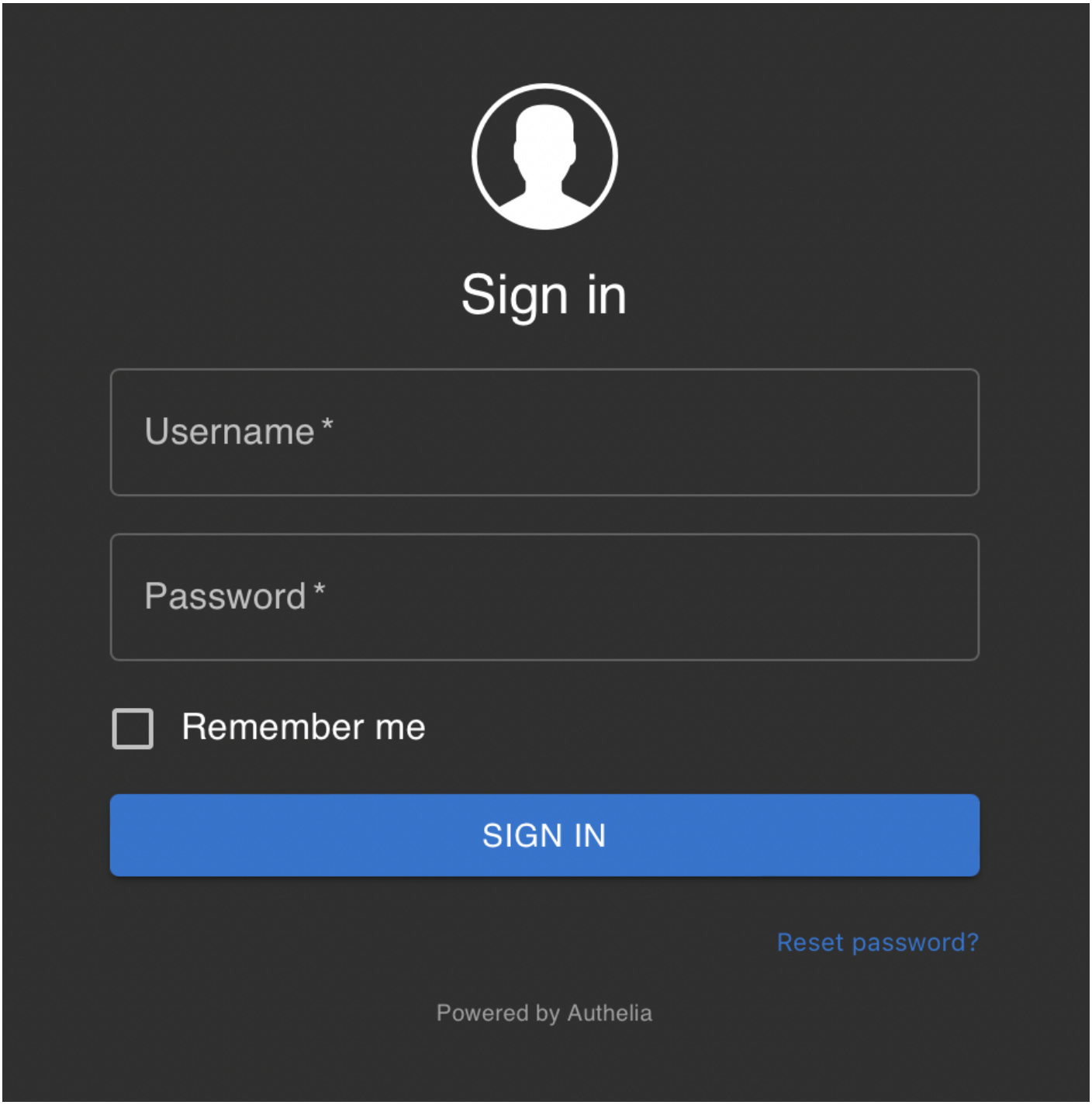
Rendez-vous dans la configuration du Proxy Host puis dans l'onglet **Advanced** et ajoutez les lignes suivantes :

```
location /authelia {
    internal;
    set $upstream_authelia http://authelia:9091/api/verify; # Adapter l'upstream en fonction de sa configuration
    proxy_pass_request_body off;
    proxy_pass $upstream_authelia;
    proxy_set_header Content-Length "";

    proxy_next_upstream error timeout invalid_header http_500 http_502 http_503;
    client_body_buffer_size 128k;
    proxy_set_header Host $host;
    proxy_set_header X-Original-URL $scheme://$http_host$request_uri;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $remote_addr;
    proxy_set_header X-Forwarded-Proto $scheme;
    proxy_set_header X-Forwarded-Host $http_host;
    proxy_set_header X-Forwarded-Uri $request_uri;
    proxy_set_header X-Forwarded-Ssl on;
    proxy_redirect  http://  $scheme://;
    proxy_http_version 1.1;
    proxy_set_header Connection "";
    proxy_cache_bypass $cookie_session;
    proxy_no_cache $cookie_session;
    proxy_buffers 4 32k;
    send_timeout 5m;
    proxy_read_timeout 240;
```

```
    proxy_send_timeout 240;
    proxy_connect_timeout 240;
}

location / {
    set $upstream_radarr http://radarr:7878; # Adapter l'upstream en fonction de sa configuration
    proxy_pass $upstream_radarr;

    auth_request /authelia;
    auth_request_set $target_url $scheme://$http_host$request_uri;
    auth_request_set $user $upstream_http_remote_user;
    auth_request_set $groups $upstream_http_remote_groups;
    proxy_set_header Remote-User $user;
    proxy_set_header Remote-Groups $groups;
    error_page 401 =302 https://auth.netfloux.com/?rd=$target_url;

    client_body_buffer_size 128k;

    proxy_next_upstream error timeout invalid_header http_500 http_502 http_503;
    send_timeout 5m;
    proxy_read_timeout 360;
    proxy_send_timeout 360;
    proxy_connect_timeout 360;
    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 X-Forwarded-Host $http_host;
    proxy_set_header X-Forwarded-Uri $request_uri;
    proxy_set_header X-Forwarded-Ssl on;
    proxy_redirect  http://  $scheme://;
    proxy_http_version 1.1;
    proxy_set_header Connection "";
    proxy_cache_bypass $cookie_session;
    proxy_no_cache $cookie_session;
    proxy_buffers 64 256k;
    set_real_ip_from 192.168.1.0/16;
    real_ip_header X-Forwarded-For;
    real_ip_recursive on;
}
```

```
# Bypass de l'auth pour l'accès à l'API
location /api {
    proxy_pass http://radarr:7878;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
}
```

Rendez-vous sur radarr.domain.com, vous devrez être redirigé sur une page de login Authelia :



Location : domain.com/radarr

Configuration d'Authelia sur Nginx

Tooling

Génération de tokens / passwords :

```
pip3 install pwgen
pwgen -1 25
quim5AhNgool9eimooceseegh
```

Initialisation Serveur Ubuntu

Commandes rapides pour initialiser un serveur Linux sous Ubuntu avec l'essentiel

Paquets de base

```
apt update && apt upgrade -y
apt install git zsh curl htop python3-pip vim wget -y
```

Shell

```
sh -c "$(curl -fsSL https://raw.github.com/ohmyzsh/ohmyzsh/master/tools/install.sh)"
git clone --depth=1 https://github.com/romkatv/powerlevel10k.git ${ZSH_CUSTOM:-$HOME/.oh-my-zsh/custom}/themes/powerlevel10k
```

Remplacer le thème par défaut de OhMyZsh par :

```
ZSH_THEME="powerlevel10k/powerlevel10k"
```

Puis recharger le shell :

```
source ~/.zshrc
```

Installation de Docker

Suivre cette documentation

Formation Linux

Formation Linux de base

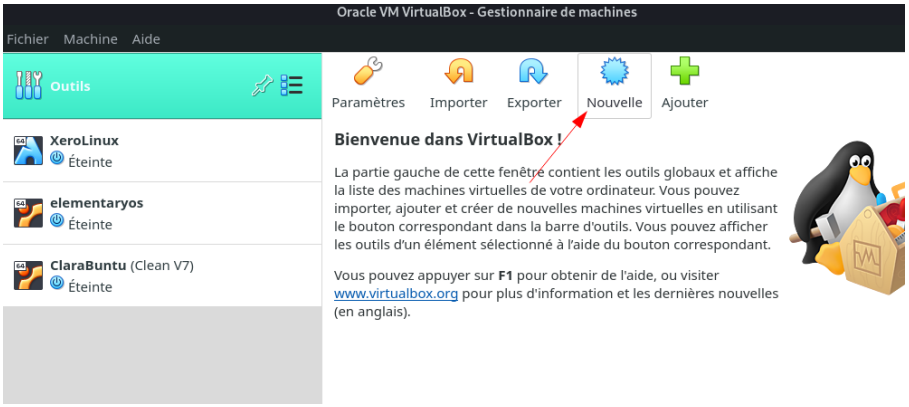
Prérequis

- Un PC sous Windows
- Une connexion Internet
- Une image ISO de Linux Ubuntu
- VirtualBox pour la virtualisation

Mise en place de l'environnement de travail

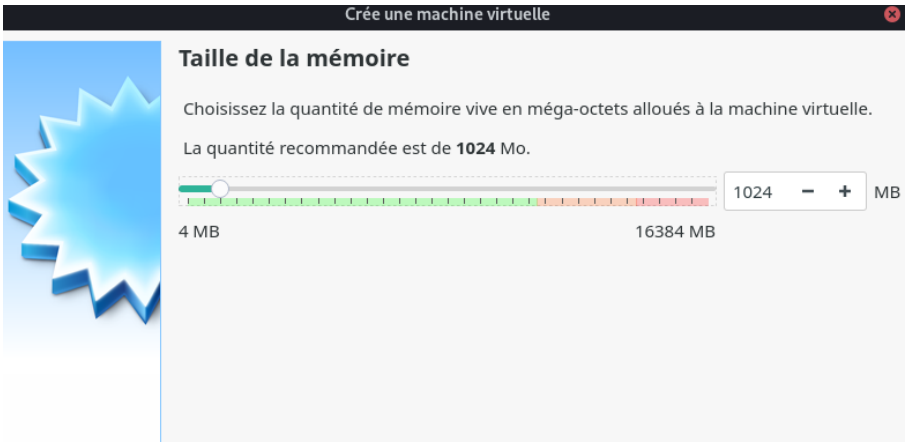
Création de la VM

Démarrer VirtualBox et créer une nouvelle machine :

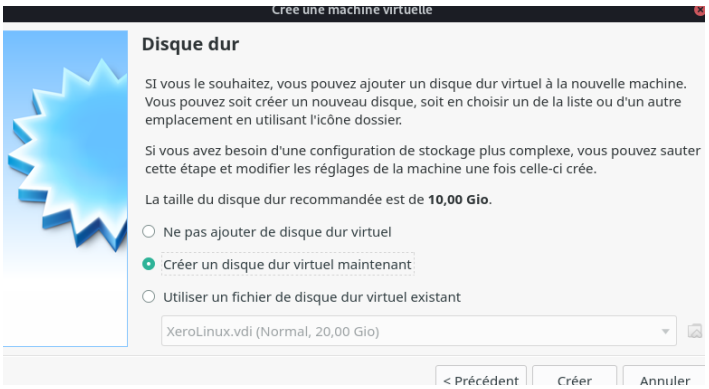


Définir son nom, emplacement et type :

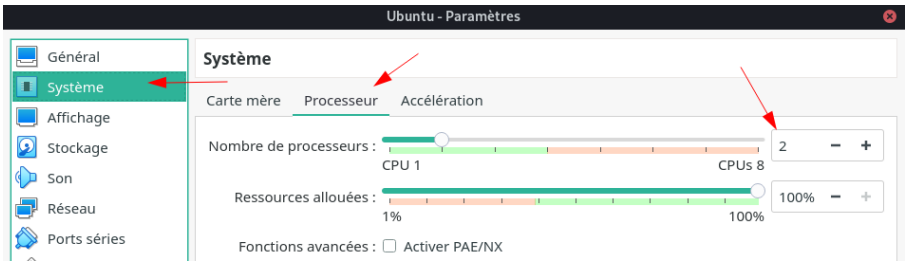
La mémoire vive allouée - 2048M recommandé :



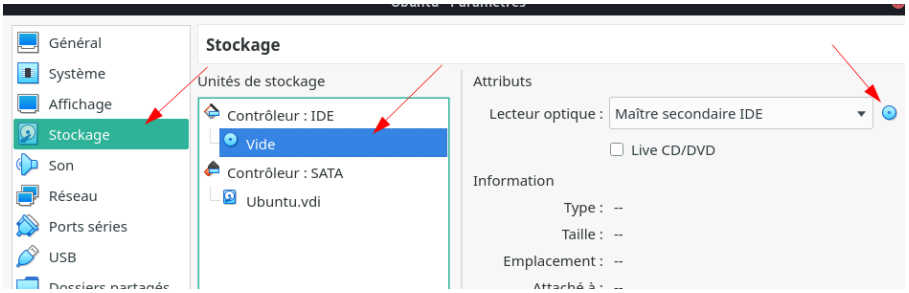
Créer un nouveau disque dur au format VDI - Taille Dynamiquement alloué de 40Go :



Ensuite, dans le menu Configuration / Système / Processeur : allouer 2 coeurs :



Dans ce même menu Configuration / Stockage > Choose a disk File et choisir l'image ISO Ubuntu téléchargée :



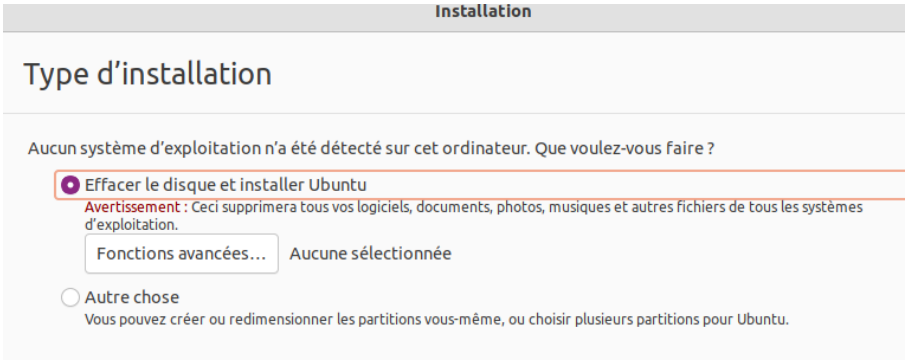
Enfin, démarrer la VM.

Installation de la VM Ubuntu

Lancer la VM et suivre l'assistant d'installation :

Choisir la disposition Clavier et Langue, puis sélectionner **l'installation normale** et cocher **Installer les logiciels tiers** puis cliquer sur continuer.

Choisir le type d'installation et cliquer **Installer maintenant**, puis **Continuer** :

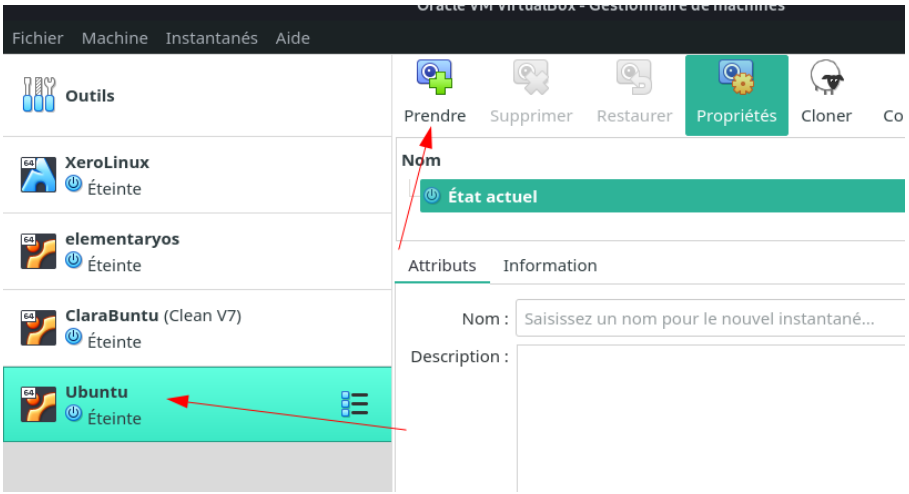


Choisir la TimeZone, et renseigner nom, prenom, mot de passe.

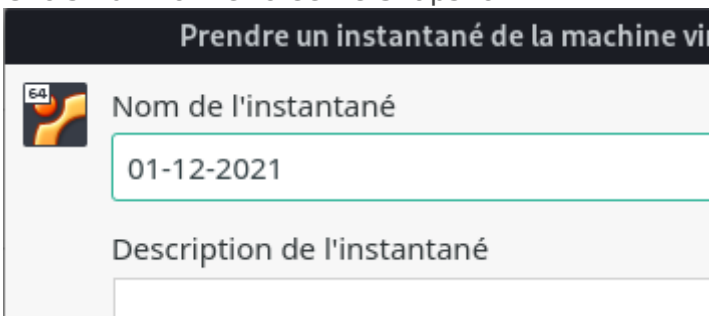
> Le clavier est potentiellement en QWERTY. Pour ne pas se louper, taper au clavier azerty (qui sera en fait qwerty) et changez le mot de passe au démarrage de la machine

Sauvegarde de l'environnement

Il est possible de sauvegarder la VM à un instant T. En cas de problème, de panne, d'erreur, on peut remettre la machine à cet état. Pour cela, arrêter la VM et effectuer un clic droit dessus :

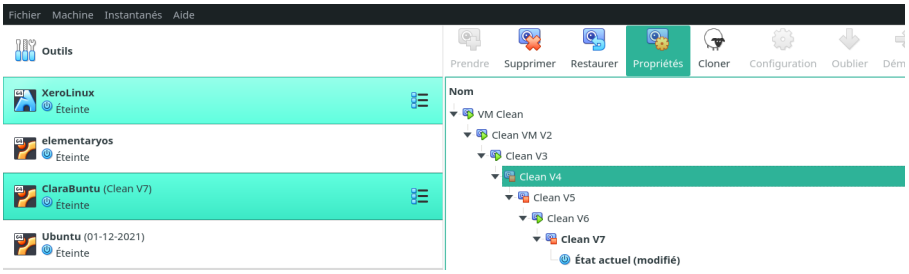


Choisir un nom et créer le Snapshot :



Restauration de l'environnement

Si besoin de restaurer la VM : l'arrêter, se rendre dans les propriétés, sélectionner la version que l'on souhaite restaurer et cliquer sur **restaurer** :




```
for i in {1..20}; do mkdir dossier_$i; done
```

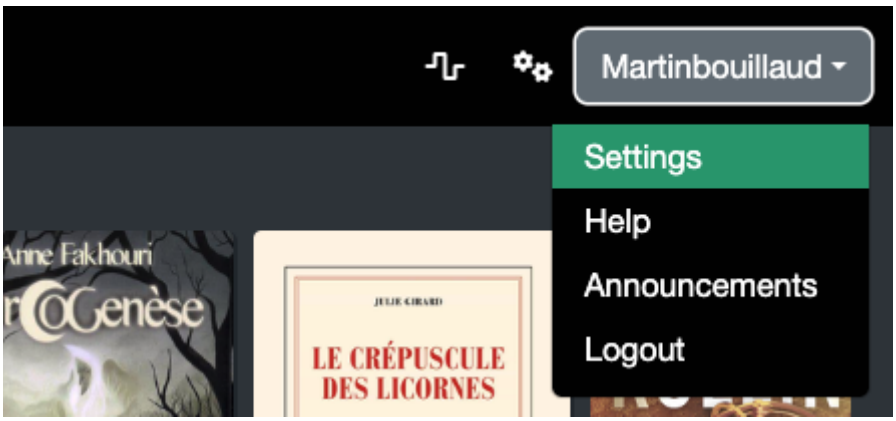
Ressources

- https://www.omgubuntu.co.uk/
- https://www.toolinux.com/
- https://linuxfr.org/

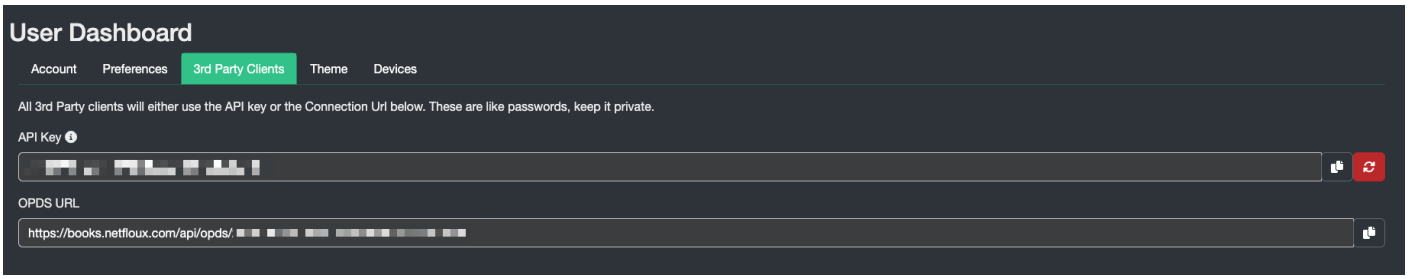
- https://www.linuxadictos.com/fr/Cat%C3%A9gorie/nouvelles
- https://google.com

Synchroniser la librairie Netfloux avec KyBook sur iOS

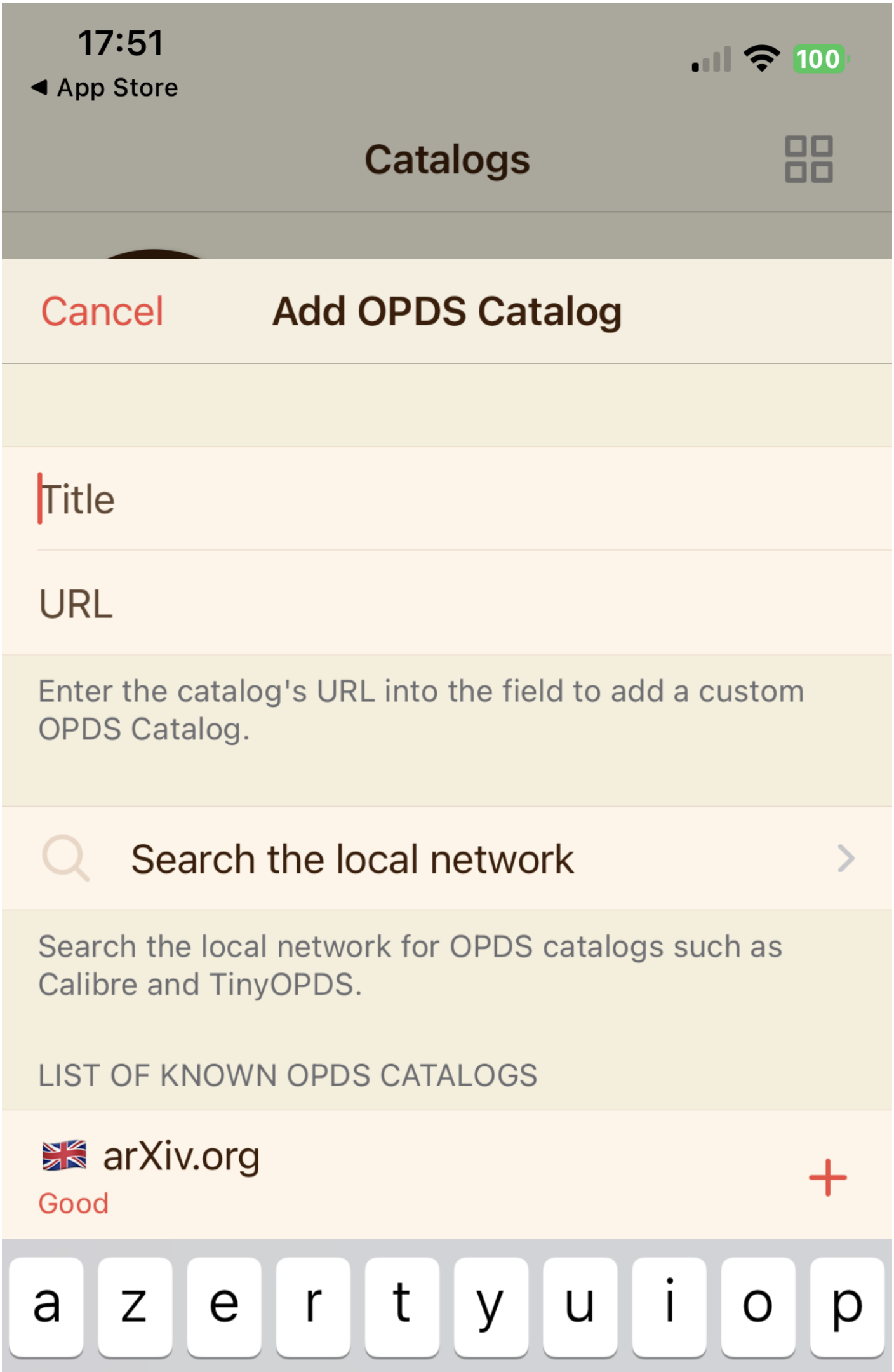
Se rendre sur books.netfloux.com, se connecter et aller dans le menu Settings en haut à droite :



Cliquer sur 3rd Party client et récupérer l'URL OPDS :



Télécharger l'application KyBook 3 et aller dans le menu Catalogs puis Add. Renseigner un nom, puis l'URL OPDS précédemment récupérée puis Terminé :


17:51
App Store
100
Catalogs
Cancel
Add OPDS Catalog
Title
URL
Enter the catalog's URL into the field to add a custom OPDS Catalog.
Search the local network
Search the local network for OPDS catalogs such as Calibre and TinyOPDS.
LIST OF KNOWN OPDS CATALOGS
arXiv.org
Good
a z e r t y u i o p

Init Serveur Ubuntu pour Web Hosting

Mise à jour du système et dépendances

```
export DEBIAN_FRONTEND=noninteractive
apt update && apt upgrade -yq
apt install -yq git zsh curl wget htop python3 bat ripgrep
```

Installation et configuration de ZSH & fzf

```
mkdir -p ~/.local/bin
ln -s /usr/bin/batcat ~/.local/bin/bat
sh -c "$(curl -fsSL https://raw.githubusercontent.com/loket/oh-my-zsh/feature/batch-mode/tools/install.sh)"
git clone --depth 1 https://github.com/junegunn/fzf.git ~/.fzf
yes | ~/.fzf/install
git clone https://github.com/zsh-users/zsh-autosuggestions ${ZSH_CUSTOM:-~/.oh-my-zsh/custom}/plugins/zsh-autosuggestions
git clone https://github.com/zsh-users/zsh-syntax-highlighting.git ${ZSH_CUSTOM:-~/.oh-my-zsh/custom}/plugins/zsh-syntax-highlighting
wget http://raw.github.com/caiogondim/bullet-train-oh-my-zsh-theme/master/bullet-train.zsh-theme -O $ZSH_CUSTOM/themes/bullet-train.zsh-theme
mv ~/.zshrc ~/.zshrc.backup
wget https://raw.githubusercontent.com/bilyboy785/public/main/zsh/zshrc.config -O ~/.zshrc
```

Installation des repos PHP et Nginx

```
DISTRIB_CODENAME=$(cat /etc/os-release | grep VERSION_CODENAME | cut -d\= -f2)
add-apt-repository ppa:ondrej/php
add-apt-repository ppa:nginx/stable

apt install -yq nginx libnginx-mod-http-geoip libnginx-mod-http-geoip2
apt install -y php8.2-apcu php8.2-bcmath php8.2-cli php8.2-common php8.2-curl php8.2-fpm php8.2-gd php8.2-gmp php8.2-igbinary php8.2-imagick php8.2-imap php8.2-intl php8.2-mbstring php8.2-memcache php8.2-memcached php8.2-msgpack php8.2-mysql php8.2-opcache php8.2-phpdbg php8.2-readline php8.2-redis php8.2-xml php8.2-zip
```

apt install -y php8.1-apcu php8.1-bcmath php8.1-cli php8.1-common php8.1-curl php8.1-fpm php8.1-gd php8.1-gmp php8.1-igbinary php8.1-imagick php8.1-imap php8.1-intl php8.1-mbstring php8.1-memcache php8.1-memcached php8.1-msgpack php8.1-mysql php8.1-opcache php8.1-phpdbg php8.1-readline php8.1-redis php8.1-xml php8.1-zip

apt install -y php8.0-apcu php8.0-bcmath php8.0-cli php8.0-common php8.0-curl php8.0-fpm php8.0-gd php8.0-gmp php8.0-igbinary php8.0-imagick php8.0-imap php8.0-intl php8.0-mbstring php8.0-memcache php8.0-memcached php8.0-msgpack php8.0-mysql php8.0-opcache php8.0-phpdbg php8.0-readline php8.0-redis php8.0-xml php8.0-zip

apt install -y php7.4-apcu php7.4-bcmath php7.4-cli php7.4-common php7.4-curl php7.4-fpm php7.4-gd php7.4-gmp php7.4-igbinary php7.4-imagick php7.4-imap php7.4-intl php7.4-mbstring php7.4-memcache php7.4-memcached php7.4-msgpack php7.4-mysql php7.4-opcache php7.4-phpdbg php7.4-readline php7.4-redis php7.4-xml php7.4-zip