# Instructions d’installation

Note : lancer toutes les commandes avec sudo

# Installer Home Assistant

1. Installer docker

|  |
| --- |
| apt update  apt upgrade  apt install docker.io docker-compose -y  systemctl enable docker |

1. Créer un fichier yaml pour installer Home Assistant

|  |
| --- |
| mkdir ~/docker  mkdir ~/docker/homeassistant  cd ~/docker/homeassistant  touch docker-compose.yml |

1. Ouvrir le fichier (gedit, nano, etc.) et copier le texte suivant

|  |
| --- |
| version: '3'  services:  homeassistant:  container\_name: homeassistant  image: ghcr.io/home-assistant/home-assistant:stable  volumes:  - ./config:/config  - /etc/localtime:/etc/localtime:ro  environment:  - TZ=America/Montreal  restart: unless-stopped  network\_mode: host |

1. Lancer l’installation via docker (note : si la commande ne fonctionne pas, essayer « docker compose up -d »)

|  |
| --- |
| docker-compose up -d |

1. Tester l’installation en accédant à la page de Home Assistant à l’adresse : <http://localhost:8123> ou http://<adresse\_IP>:8123. On devrait voir la page suivante :

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Le contenu généré par l’IA peut être incorrect.

1. Par la suite créer un compte utilisateur et connecter les prises intelligentes. Le nom donné aux prises doit respecter le format suivant « asset\_XXXXX », où « XXXXX » représente l’asset tag dans Snipe-IT.

# Installer Snipe-IT

Procédure adaptée de <https://www.youtube.com/watch?v=nuHrlFCmu-U&t=0s>

1. Créer un fichier yaml pour installer Snipe-IT

|  |
| --- |
| mkdir ~/docker/snipe-it  cd ~/docker/snipe-it  touch docker-compose.yml |

1. Ouvrir le fichier (gedit, nano, etc.) et copier le texte suivant

|  |
| --- |
| # Compose file for production.  volumes:  db\_data:  storage:  services:  app:  image: snipe/snipe-it:v8.0.4  restart: unless-stopped  volumes:  - storage:/var/lib/snipeit  ports:  - "8000:80"  depends\_on:  db:  condition: service\_healthy  env\_file:  - .env  db:  image: mariadb:11.5.2  restart: unless-stopped  volumes:  - db\_data:/var/lib/mysql  environment:  MYSQL\_DATABASE: snipeit  MYSQL\_USER: snipeit  MYSQL\_PASSWORD: ele795  MYSQL\_ROOT\_PASSWORD: ele795  healthcheck:  # https://mariadb.com/kb/en/using-healthcheck-sh/#compose-file-example  test: ["CMD", "healthcheck.sh", "--connect", "--innodb\_initialized"]  interval: 5s  timeout: 1s  retries: 5 |

1. Créer un fichier .env et copier le texte suivant

|  |
| --- |
| # --------------------------------------------  # REQUIRED: DOCKER SPECIFIC SETTINGS  # --------------------------------------------  APP\_VERSION=v8.0.4  APP\_PORT=8000  # --------------------------------------------  # REQUIRED: BASIC APP SETTINGS  # --------------------------------------------  APP\_ENV=production  APP\_DEBUG=false  # Please regenerate the APP\_KEY value by calling `docker compose run --rm app php artisan key:generate --show`. Copy paste the value here  APP\_KEY=<APP\_KEY>  APP\_URL=http://<IP>:8000  # https://en.wikipedia.org/wiki/List\_of\_tz\_database\_time\_zones - TZ identifier  APP\_TIMEZONE=America/Montreal  APP\_LOCALE=en-US  MAX\_RESULTS=500  # --------------------------------------------  # REQUIRED: UPLOADED FILE STORAGE SETTINGS  # --------------------------------------------  PRIVATE\_FILESYSTEM\_DISK=local  PUBLIC\_FILESYSTEM\_DISK=local\_public  # --------------------------------------------  # REQUIRED: DATABASE SETTINGS  # --------------------------------------------  DB\_CONNECTION=mysql  DB\_HOST=db  DB\_PORT='3306'  DB\_DATABASE=snipeit  DB\_USERNAME=snipeit  DB\_PASSWORD=ele795  MYSQL\_ROOT\_PASSWORD=ele795  DB\_PREFIX=null  DB\_DUMP\_PATH='/usr/bin'  DB\_CHARSET=utf8mb4  DB\_COLLATION=utf8mb4\_unicode\_ci  # --------------------------------------------  # OPTIONAL: SSL DATABASE SETTINGS  # --------------------------------------------  DB\_SSL=false  DB\_SSL\_IS\_PAAS=false  DB\_SSL\_KEY\_PATH=null  DB\_SSL\_CERT\_PATH=null  DB\_SSL\_CA\_PATH=null  DB\_SSL\_CIPHER=null  DB\_SSL\_VERIFY\_SERVER=null  # --------------------------------------------  # REQUIRED: OUTGOING MAIL SERVER SETTINGS  # --------------------------------------------  MAIL\_MAILER=smtp  MAIL\_HOST=smtp.gmail.com  MAIL\_PORT=587  MAIL\_USERNAME=snipe.it.lacime@gmail.com  MAIL\_PASSWORD=<APP\_PASSWORD>  MAIL\_TLS\_VERIFY\_PEER=true  MAIL\_FROM\_ADDR=snipe.it.lacime@gmail.com  MAIL\_FROM\_NAME='Snipe-IT'  MAIL\_REPLYTO\_ADDR=snipe.it.lacime@gmail.com  MAIL\_REPLYTO\_NAME='Snipe-IT'  MAIL\_AUTO\_EMBED\_METHOD='attachment'  # --------------------------------------------  # REQUIRED: DATA PROTECTION  # --------------------------------------------  ALLOW\_BACKUP\_DELETE=false  ALLOW\_DATA\_PURGE=false  # --------------------------------------------  # REQUIRED: IMAGE LIBRARY  # This should be gd or imagick  # --------------------------------------------  IMAGE\_LIB=gd  # --------------------------------------------  # OPTIONAL: BACKUP SETTINGS  # --------------------------------------------  MAIL\_BACKUP\_NOTIFICATION\_DRIVER=null  MAIL\_BACKUP\_NOTIFICATION\_ADDRESS=null  BACKUP\_ENV=true  # --------------------------------------------  # OPTIONAL: SESSION SETTINGS  # --------------------------------------------  SESSION\_LIFETIME=12000  EXPIRE\_ON\_CLOSE=false  ENCRYPT=false  COOKIE\_NAME=snipeit\_session  COOKIE\_DOMAIN=null  SECURE\_COOKIES=false  API\_TOKEN\_EXPIRATION\_YEARS=40  # --------------------------------------------  # OPTIONAL: SECURITY HEADER SETTINGS  # --------------------------------------------  #APP\_TRUSTED\_PROXIES=192.168.1.1,10.0.0.1,172.16.0.0/12  ALLOW\_IFRAMING=false  REFERRER\_POLICY=same-origin  ENABLE\_CSP=false  CORS\_ALLOWED\_ORIGINS=null  ENABLE\_HSTS=false  # --------------------------------------------  # OPTIONAL: CACHE SETTINGS  # --------------------------------------------  CACHE\_DRIVER=file  SESSION\_DRIVER=file  QUEUE\_DRIVER=sync  CACHE\_PREFIX=snipeit  # --------------------------------------------  # OPTIONAL: REDIS SETTINGS  # --------------------------------------------  REDIS\_HOST=null  REDIS\_PASSWORD=null  REDIS\_PORT=6379  # --------------------------------------------  # OPTIONAL: MEMCACHED SETTINGS  # --------------------------------------------  MEMCACHED\_HOST=null  MEMCACHED\_PORT=null  # --------------------------------------------  # OPTIONAL: PUBLIC S3 Settings  # --------------------------------------------  PUBLIC\_AWS\_SECRET\_ACCESS\_KEY=null  PUBLIC\_AWS\_ACCESS\_KEY\_ID=null  PUBLIC\_AWS\_DEFAULT\_REGION=null  PUBLIC\_AWS\_BUCKET=null  PUBLIC\_AWS\_URL=null  PUBLIC\_AWS\_BUCKET\_ROOT=null  # --------------------------------------------  # OPTIONAL: PRIVATE S3 Settings  # --------------------------------------------  PRIVATE\_AWS\_ACCESS\_KEY\_ID=null  PRIVATE\_AWS\_SECRET\_ACCESS\_KEY=null  PRIVATE\_AWS\_DEFAULT\_REGION=null  PRIVATE\_AWS\_BUCKET=null  PRIVATE\_AWS\_URL=null  PRIVATE\_AWS\_BUCKET\_ROOT=null  # --------------------------------------------  # OPTIONAL: AWS Settings  # --------------------------------------------  AWS\_ACCESS\_KEY\_ID=null  AWS\_SECRET\_ACCESS\_KEY=null  AWS\_DEFAULT\_REGION=null  # --------------------------------------------  # OPTIONAL: LOGIN THROTTLING  # --------------------------------------------  LOGIN\_MAX\_ATTEMPTS=5  LOGIN\_LOCKOUT\_DURATION=60  RESET\_PASSWORD\_LINK\_EXPIRES=900  # --------------------------------------------  # OPTIONAL: MISC  # --------------------------------------------  LOG\_CHANNEL=stderr  LOG\_MAX\_DAYS=10  APP\_LOCKED=false  APP\_CIPHER=AES-256-CBC  APP\_FORCE\_TLS=false  GOOGLE\_MAPS\_API=  LDAP\_MEM\_LIM=500M  LDAP\_TIME\_LIM=600 |

1. Remplacer les éléments surlignés en jaune

|  |  |
| --- | --- |
| <APP\_KEY> | Il faut générer une clé avec la commande « *docker compose run --rm app php artisan key:generate –show* » et coller le résultat ici. |
| <IP> | Remplacer par l’adresse IP de la machine. |
| MAIL\_HOST  MAIL\_USERNAME  MAIL\_FROM\_ADDR  MAIL\_REPLYTO\_ADDR | Serveur de la boîte de courriel de Snipe-IT. Ici on utilise une adresse courriel google comme exemple. |
| MAIL\_PASSWORD | Mot de passe d’application (*app password*) pour pouvoir envoyer des courriels. Ici on l’a généré à partir de google. |

1. Modifier les permissions du fichier .env

|  |
| --- |
| chmod 400 .env |

1. Tester l’installation en accédant à la page de Snipe-IT à l’adresse : <http://localhost:8000> ou avec l’adresse IP.

Une image contenant texte, logiciel, Icône d’ordinateur, Page web

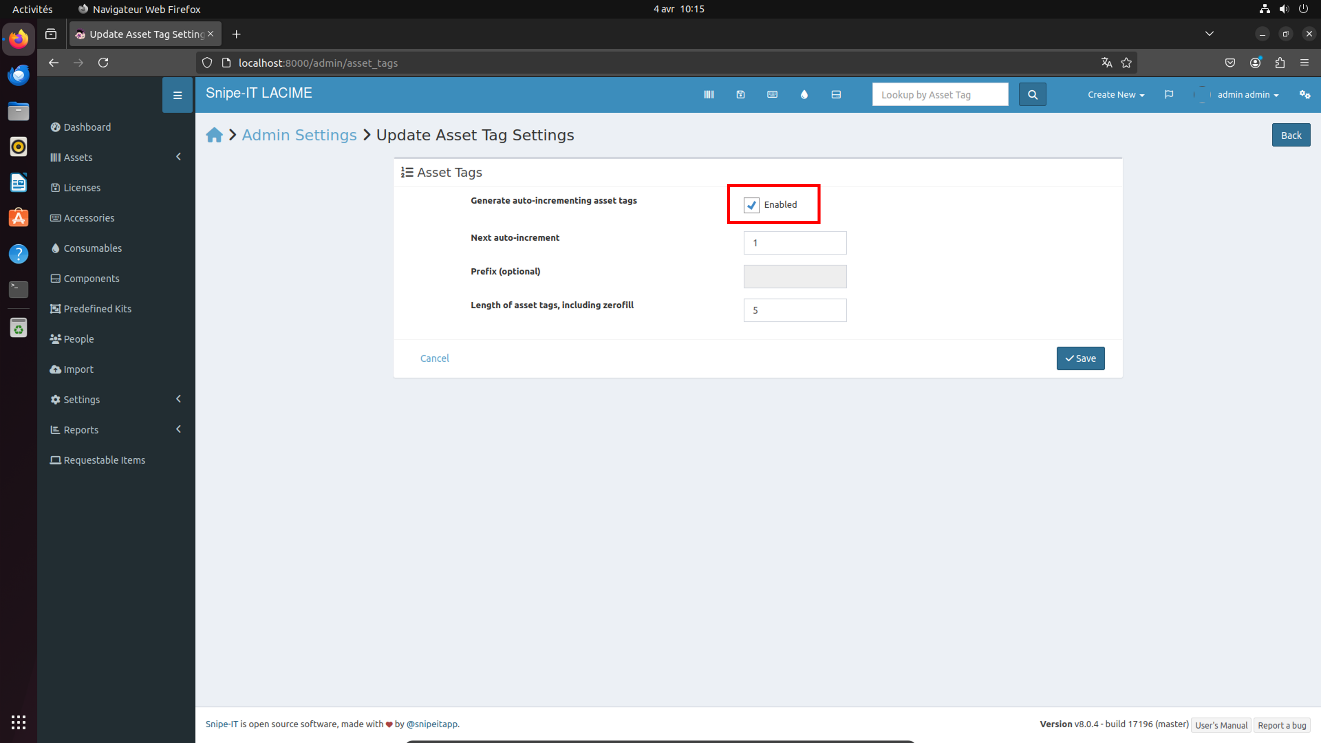
Le contenu généré par l’IA peut être incorrect.

1. Créer un compte administrateur. À partir de là, on peut personnaliser l’interface, configurer Snipe-IT et ajouter des assets. Pour ce faire, il faut d’abord créer une catégorie (ex : oscilloscopes), puis un modèle.

Une image contenant texte, capture d’écran, logiciel, Icône d’ordinateur

Le contenu généré par l’IA peut être incorrect.

1. S’assurer que les asset tags sont en mode auto incrémental



# Script

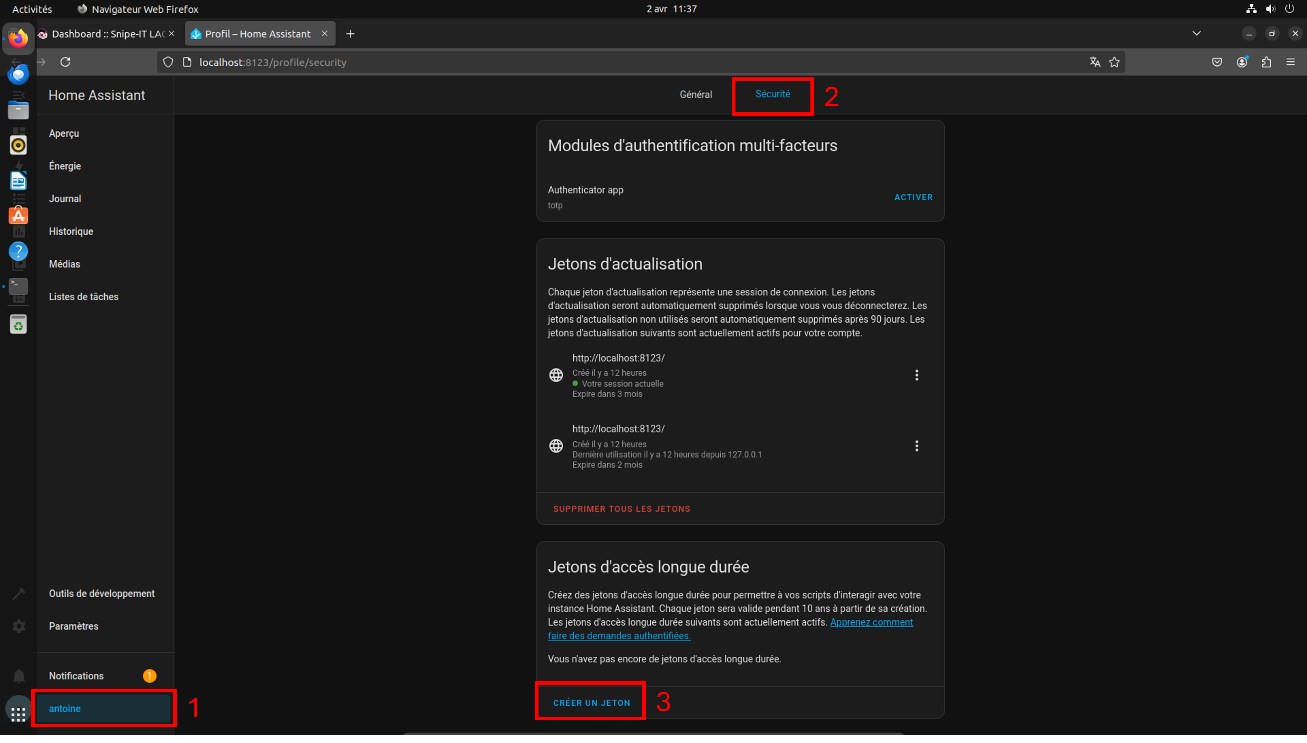
1. Créer un script python dans ~/docker/homeassistant

|  |
| --- |
| cd ~/docker/homeassistant  touch webhook\_snipeit\_to\_homeassistant.py |

1. Ouvrir le fichier et copier le texte suivant

|  |
| --- |
| from fastapi import FastAPI, Request  import aiohttp  import uvicorn  import re  # === CONFIGURATION ===  HA\_URL = "http://localhost:8123"  HA\_TOKEN = "<API\_TOKEN>"  # États cibles  STATE\_ON = "checked out"  STATE\_OFF = "checked in"  # === APP FASTAPI ===  app = FastAPI()  async def set\_switch(state: str, id: str):  url = f"{HA\_URL}/api/services/switch/turn\_{state}"  headers = {  "Authorization": f"Bearer {HA\_TOKEN}",  "Content-Type": "application/json"  }  payload = {"entity\_id": f"switch.asset\_{id}"}  print(f"switch.asset\_{id}")  async with aiohttp.ClientSession() as session:  async with session.post(url, headers=headers, json=payload) as response:  response.raise\_for\_status()  print(f"[HA] switch.switch.asset\_{id} -> {state}")  @app.post("/webhook")  async def receive\_webhook(request: Request):  payload = await request.json()  #print(f"[Webhook reçu] ID : {payload}")  try:  title\_link = payload["attachments"][0]["title\_link"] # ex: http://.../hardware/1  match = re.search(r"/hardware/(\d+)", title\_link)  if match:  hardware\_id = match.group(1).zfill(5)  print(f"[INFO] Hardware ID : {hardware\_id}")  if payload.get("text", {}).find(STATE\_ON) != -1:  await set\_switch("on", hardware\_id)  print("[Webhook reçu] Nouveau statut : checked out")  elif payload.get("text", {}).find(STATE\_OFF) != -1:  await set\_switch("off", hardware\_id)  print("[Webhook reçu] Nouveau statut : checked in")  else:  print("[INFO] Statut ignoré")  else:  print("[ERREUR] ID non trouvé dans title\_link")  return {"status": "erreur"}  except Exception as e:  print(f"[ERREUR] Webhook mal formé ou données manquantes : {e}")  return {"status": "ok"} |

1. Générer un token dans Home Assistant et le remplacer dans le script



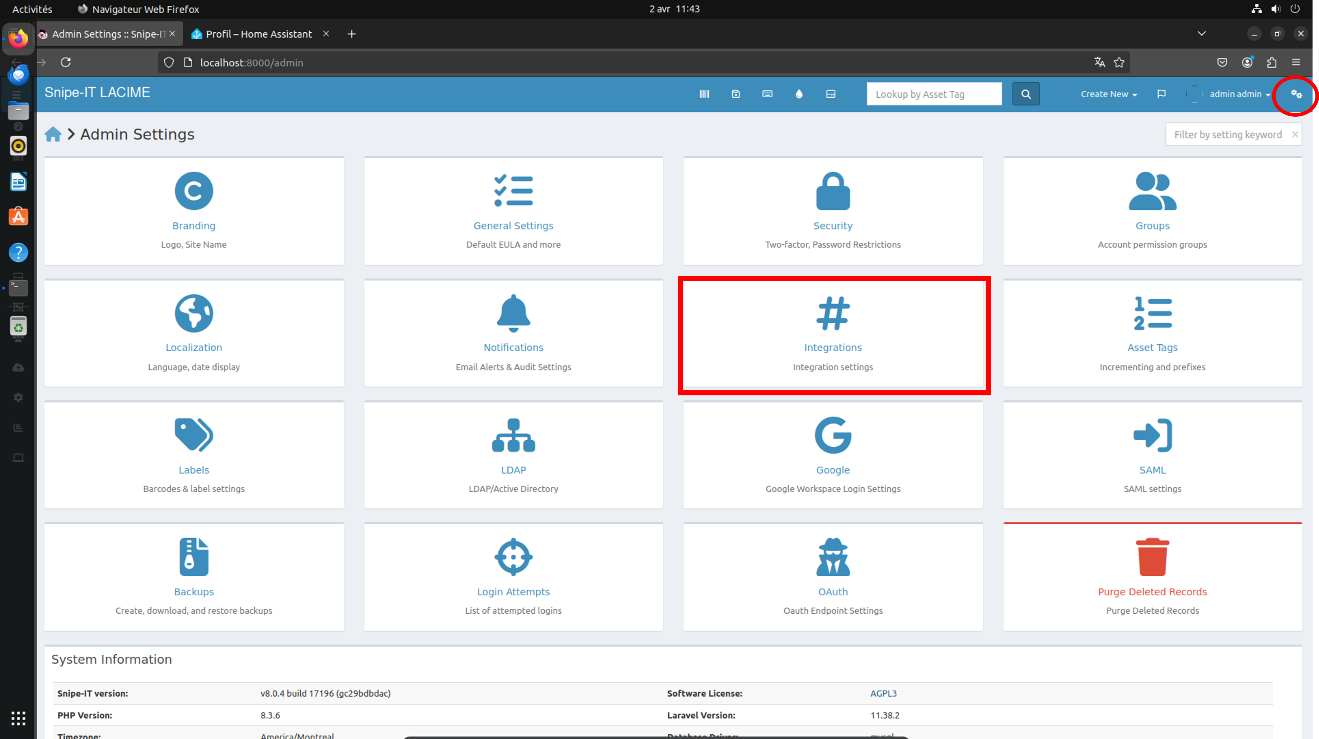
1. Installer uvicorn, fastapi et aiohttp

|  |
| --- |
| apt install -y python3  apt install -y python3-pip  pip install fastapi uvicorn aiohttp |

1. Lancer le script

|  |
| --- |
| uvicorn webhook\_snipeit\_to\_homeassistant:app --host <IP> --port 8080 |

1. Créer un webhook dans Snipe-IT



Une image contenant texte, capture d’écran, logiciel, Icône d’ordinateur

Le contenu généré par l’IA peut être incorrect.

1. S’assurer qu’il n’y a pas d’erreur et cliquer sur « Save ». À ce stade-ci, tout devrait fonctionner. Vérifier que lorsqu’on clique sur « Checkout » la prise s’allume, et que lorsqu’on clique sur « Checkin », elle s’éteint. On devrait voir ce résultat dans le terminal :

Une image contenant texte, capture d’écran, Police, logiciel

Le contenu généré par l’IA peut être incorrect.

1. Une fois qu’on a validé le fonctionnement, on peut créer un service pour que le script soit lancé automatiquement au démarrage. Créer un fichier dans /etc/systemd/system

|  |
| --- |
| touch /etc/systemd/system/snipeit\_to\_ha.service |

1. Copier le texte suivant, en remplaçant <IP> par votre adresse IP

|  |
| --- |
| [Unit]  Description=Link Snipe-IT inventory with Home Assistant  After=network.target  [Service]  ExecStartPre=/bin/sleep 30  User=root  Group=root  WorkingDirectory=/root/docker/homeassistant/  ExecStart=uvicorn webhook\_snipeit\_to\_homeassistant:app --port 8080 --host <IP>  Restart=on-failure  PrivateTmp=true  [Install]  WantedBy=multi-user.target |

1. Activer le service et vérifier qu’il fonctionne après un redémarrage

|  |
| --- |
| systemctl enable snipeit\_to\_ha |