AWX direkt auf der Maschine

Hier eine komplette, kopierfertige Anleitung für die native Installation von AWX 17.1.0 auf Ubuntu 24.04 ohne Docker, mit bestehendem Nginx Proxy Manager als Reverse-Proxy. Einfach der Reihe nach abarbeiten:

---

1. System-Pakete installieren

```
sudo apt update
sudo apt install -y \
  git build-essential libssl-dev libffi-dev python3-venv python3-pip \
  libpq-dev postgresql postgresql-contrib redis-server rabbitmq-server \
  nginx curl pwgen
```

---

2. PostgreSQL-Datenbank & RabbitMQ anlegen

a) PostgreSQL

```
sudo -u postgres psql <<'EOF'
CREATE USER awx WITH PASSWORD 'AWX_db_passwort';
CREATE DATABASE awx OWNER awx ENCODING 'UTF8';
\q
EOF
```

b) RabbitMQ

```
sudo rabbitmqctl add_user awx MQ_passwort
sudo rabbitmqctl set_user_tags awx administrator
sudo rabbitmqctl set_permissions -p / awx ".*" ".*" ".*"
```

---

3. AWX-Quellcode auschecken

```
sudo mkdir -p /opt/awx
sudo chown $USER:$USER /opt/awx
cd /opt/awx
git clone -b 17.1.0 https://github.com/ansible/awx.git .
```

---

4. Python-Virtualenv & Abhängigkeiten

```
cd /opt/awx
python3 -m venv .venv
source .venv/bin/activate
pip install --upgrade pip
pip install -r requirements/requirements.txt
pip install psycopg2-binary redis kombu django gunicorn
deactivate
```

---

5. AWX-Settings anlegen

Erstelle `~/awx_settings.py` mit:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'awx',
```

```
        'USER': 'awx',
        'PASSWORD': 'AWX_db_passwort',
        'HOST': 'localhost',
        'PORT': '5432',
    }
}

BROKER_URL = 'amqp://awx:MQ_passwort@localhost:5672//'

CACHES = {
    'default': {
        'BACKEND': 'django_redis.cache.RedisCache',
        'LOCATION': 'redis://127.0.0.1:6379/1',
        'OPTIONS': {'CLIENT_CLASS': 'django_redis.client.DefaultClient'},
    }
}

SECRET_KEY = '$(pwgen -N1 -s32)'
ALLOWED_HOSTS = ['*']
```

Exportiere:

```
export AWX_SETTINGS_MODULE=awx_settings
```

---

6. Datenbank-Migration & Admin anlegen

```
cd /opt/awx/awx
source /opt/awx/.venv/bin/activate
./manage.py makemigrations
./manage.py migrate
./manage.py createsuperuser --username=admin --email=admin@example.com
deactivate
```

---

7. Gunicorn als systemd-Service

Erstelle `/etc/systemd/system/awx-gunicorn.service`:

```
[Unit]
Description=AWX Gunicorn Service
After=network.target

[Service]
User=www-data
Group=www-data
WorkingDirectory=/opt/awx/awx
Environment="AWX_SETTINGS_MODULE=awx_settings"
Environment="PATH=/opt/awx/.venv/bin"
ExecStart=/opt/awx/.venv/bin/gunicorn \
    --access-logfile - --workers 3 --bind 127.0.0.1:8052 awx.wsgi:application

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

Aktivieren und starten:

```
sudo systemctl daemon-reload
sudo systemctl enable --now awx-gunicorn
```

---

8. Statische Dateien sammeln

```
cd /opt/awx/awx
source /opt/awx/.venv/bin/activate
./manage.py collectstatic --noinput
deactivate
```

Die Assets liegen nun in `/opt/awx/awx/ui/static/`.

---

9. Nginx Proxy Manager konfigurieren

Im NPM-WebUI:

1. **Proxy Host** anlegen
   - **Domain Names**: `awx.deine-domain.tld`
   - **Scheme**: `http`
   - **Forward Hostname/IP**: `127.0.0.1`
   - **Forward Port**: `8052`
2. **Advanced → Custom Nginx Configuration**

   ```
   location /static/ {
       alias /opt/awx/awx/ui/static/;
       access_log off;
       expires max;
   }
   ```

3. **SSL** (optional): Let’s Encrypt aktivieren, “Block Common Exploits” ankreuzen.

Speichern und testen.

---

10. Fertig & Zugriff

Rufe im Browser auf:

```
https://awx.deine-domain.tld/
```

Login mit `admin` und deinem Passwort.

—
Viel Erfolg!

Revision #1
Created 18 June 2025 11:44:10 by Stefan Mechler
Updated 18 June 2025 11:44:21 by Stefan Mechler