Hard Techno Agent — End■to■End Playbook

> Version 1.0 — Generated: now
> Scope: n8n + Python (FastAPI) + optional Traefik/Redis/Postgres + Human■in■the■Loop Rating + daily cadence.

0) Ziele & Überblick

Ziel: Ein Agent, der aus aktuellen Techno■Quellen **Signale** sammelt (Charts, Sets, Blogs, YouTube), daraus **Prompts/Parameter** ableitet, **täglich 10–20 Tracks generiert** (z. B. via Suno■API), **Stems** (optional) extrahiert und durch **dein Feedback** (■/■/■ + Stem■Keep/Remove) **lernt**, was für dich funktioniert (Buckets: *hard■146*, *peak■138*, …).
Architektur■Skizze (Text):
`Sources → n8n workflows → pyapi (FastAPI) → Gen (Suno/API) → Storage (WAV+STEMS) → Rating UI (Telegram/Webhook) → Learn (weights)`
Tagesrhythmus:
- **00:00–12:00** generieren (max 10–20)
- **12:00–24:00** bewerten (Rating & Stems)
- **23:30** lernen & reporten

1) Voraussetzungen

- Azure■Abo + VM (Ubuntu 22.04 LTS, x64)
- SSH■Key (.pem) auf dem Mac, IP■basierte NSG■Regel (Port 22 nur für deine IPv4)
- Docker & docker■compose **auf der VM**
- Optional: Domain + Traefik (HTTPS)
- Optional: Postgres & Redis (Queue■Mode)

2) Azure VM — Schritt für Schritt

1. **VM anlegen (Portal)**
   - *Resource group*: `rg-n8n`
   - *Region*: Germany West Central (oder West Europe)
   - *Image*: **Ubuntu Server 22.04 LTS (Canonical)**
   - *Size*: **B2ms** (2 vCPU/8 GB, günstig) **oder** **D2as_v5** (2/8, stabil)
   - *Auth*: SSH■Key erstellen **oder** bestehenden Key nutzen
   - *Public IP*: Static, SKU Standard
   - *NSG (Firewall)*: Inbound Allow **80, 443**, **22 nur von eigener IP** (z. B. `84.172.60.202/32`)
2. **SSH**
   ```
   ssh -i ~/.ssh/vm-n8n_key.pem azureuser@
   ```
3. **System updaten**
   ```
   sudo apt update && sudo apt upgrade -y
   sudo reboot
   ```
4. **Docker & compose**
   ```
   curl -fsSL https://get.docker.com | sh
   sudo apt install -y docker-compose-plugin
   sudo usermod -aG docker $USER
   exit
   # neu einloggen
   ssh -i ~/.ssh/vm-n8n_key.pem azureuser@
   docker ps
   ```

3) Option A (empfohlen): IP■Variante + SSH■Tunnel (ohne Traefik)

```
### 3.1 Projektordner & Compose
mkdir -p ~/n8n && cd ~/n8n
cat > docker-compose.yml << 'EOF'
version: "3.8"
services:
  n8n:
    image: n8nio/n8n:latest
    container_name: n8n
    ports:
      - "5678:5678"   # Zugriff nur via SSH-Tunnel
    environment:
      - N8N_HOST=
      - N8N_PORT=5678
      - N8N_PROTOCOL=http
      - WEBHOOK_URL=http:///
      - N8N_BASIC_AUTH_ACTIVE=true
      - N8N_BASIC_AUTH_USER=admin
      - N8N_BASIC_AUTH_PASSWORD=supersecret
      - N8N_ENCRYPTION_KEY=PUT_A_LONG_RANDOM_KEY_HERE
      - TZ=Europe/Berlin
    volumes:
      - n8n_data:/home/node/.n8n
    restart: unless-stopped
  pyapi:
    build: ./python-svc
    container_name: pyapi
    expose:
      - "8000"      # nur intern
    restart: unless-stopped
volumes:
  n8n_data:
EOF
### 3.2 Python■Service (FastAPI)
mkdir -p ~/n8n/python-svc && cd ~/n8n/python-svc
cat > requirements.txt << 'EOF'
fastapi==0.115.0
uvicorn[standard]==0.30.0
requests==2.32.3
beautifulsoup4==4.12.3
EOF
cat > main.py << 'EOF'
from fastapi import FastAPI, Query, Body
from pydantic import BaseModel
from typing import Dict
import requests, re, time
from bs4 import BeautifulSoup
app = FastAPI()
PROMPTS: Dict[str, dict] = {}
@app.get("/health")
def health():
```

```
    return {"status": "ok"}
@app.post("/hello")
def hello(name: str = "Sid"):
    return {"msg": f"Hello from Python, {name} ■"}
@app.get("/beatport_top")
def beatport_top(genre: str = Query("techno", pattern="^(techno|hard-techno)$")):
    url_map = {
        "techno": "https://www.beatport.com/genre/techno-peak-time-driving/6/top-100?per-page=100",
        "hard-techno": "https://www.beatport.com/genre/hard-techno/8/top-100?per-page=100",
    }
    url = url_map[genre]
    html = requests.get(url, timeout=20).text
    soup = BeautifulSoup(html, "html.parser")
    titles = [e.get_text(strip=True) for e in soup.select(".chart-track .buk-track-primary-title")]
    artists = [e.get_text(strip=True).replace("Artists ", "") for e in soup.select(".chart-track .buk-track-artists")]
    links = []
    for a in soup.select(".chart-track a.buk-track-link"):
        href = a.get("href", "")
        if href and "/track/" in href:
            links.append("https://www.beatport.com" + href)
    if not titles or not artists or not links:
        # Fallback
        links = list(dict.fromkeys(re.findall(r'href="(/track/[^"#?]+/\d+)"', html)))
        links = ["https://www.beatport.com"+l for l in links]
    n = min(len(titles), len(artists), len(links), 100)
    out = [{
        "source":"beatport","subsource":genre,
        "title": titles[i] if i < len(titles) else "",
        "artist": artists[i] if i < len(artists) else "",
        "url": links[i]
    } for i in range(n)]
    return {"count": len(out), "items": out}
class RatingIn(BaseModel):
    prompt_id: str
    rating: int  # -1,0,+1
@app.post("/update_prompt_weight")
def update_prompt_weight(data: RatingIn):
    p = PROMPTS.setdefault(data.prompt_id, {"weight":1.0,"last_score":0.0,"wins":0,"losses":0})
    alpha = 0.3
    p["last_score"] = (1-alpha)*p["last_score"] + alpha*data.rating
    if data.rating > 0: p["wins"] += 1
    if data.rating < 0: p["losses"] += 1
    p["weight"] = max(0.1, min(5.0, 1.0 + p["last_score"]))
    return {"ok": True, "prompt": p}
@app.get("/sample_prompts")
def sample_prompts(bucket: str = "hard-146", k: int = 5):
    import random
    candidates = [
        {"id":"p1","bucket":"hard-146","text":"Hard Techno 146 BPM, relentless kick, industrial percs, rave stabs,
short breaks.","weight":PROMPTS.get("p1",{}).get("weight",1.0)},
        {"id":"p2","bucket":"hard-146","text":"Schranz groove, 146 BPM, pounding low end, metallic hats,
minimal melody.","weight":PROMPTS.get("p2",{}).get("weight",1.0)},
```

```
        {"id":"p3","bucket":"hard-146","text":"Ravey acid stabs, 146 BPM, distortion tastefully
controlled.","weight":PROMPTS.get("p3",{}).get("weight",1.0)},
        {"id":"p4","bucket":"hard-146","text":"Industrial warehouse vibe, 146 BPM, big room energy, short
risers.","weight":PROMPTS.get("p4",{}).get("weight",1.0)},
    ]
    pool = [c for c in candidates if c["bucket"] == bucket]
    weights = [max(0.1,c["weight"]) for c in pool]
    picks = random.choices(pool, weights=weights, k=min(k,len(pool)))
    return {"items": picks, "ts": int(time.time())}
class StemChoice(BaseModel):
    gen_track_id: str
    keep: Dict[str, bool]  # {"drums": True, "bass": True, ...}
@app.post("/save_stems_choice")
def save_stems_choice(data: StemChoice):
    # Hier würdest du persistieren (DB)
    return {"ok": True, "received": data.keep}
EOF
cat > Dockerfile << 'EOF'
FROM python:3.11-slim
WORKDIR /app
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt
COPY . .
CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8000"]
EOF
### 3.3 Stack starten
cd ~/n8n
docker compose up -d --build
docker ps
### 3.4 n8n sicher erreichen (SSH■Tunnel)
Auf dem Mac in **einem neuen Terminal**:
ssh -i ~/.ssh/vm-n8n_key.pem -L 5678:localhost:5678 azureuser@
Browser: `http://localhost:5678` → Owner■Account anlegen → Login.
```

4) Option B: HTTPS + Traefik (Domain)

4.1 DNS
A■Record setzen: `n8n.deinedomain.de → `
4.2 Compose (Traefik + n8n + pyapi)
```
cat > ~/n8n/docker-compose-traefik.yml << 'EOF'
version: "3.8"
services:
  traefik:
    image: traefik:v3.0
    command:
      - "--providers.docker=true"
      - "--entrypoints.web.address=:80"
      - "--entrypoints.websecure.address=:443"
      - "--certificatesresolvers.le.acme.tlschallenge=true"
      - "--certificatesresolvers.le.acme.email=you@example.com"
      - "--certificatesresolvers.le.acme.storage=/letsencrypt/acme.json"
    ports: [ "80:80", "443:443" ]
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock:ro
      - traefik_letsencrypt:/letsencrypt
    restart: unless-stopped
  n8n:
    image: n8nio/n8n:latest
    environment:
      - N8N_HOST=n8n.deinedomain.de
      - N8N_PORT=5678
      - N8N_PROTOCOL=https
      - WEBHOOK_URL=https://n8n.deinedomain.de/
      - N8N_BASIC_AUTH_ACTIVE=true
      - N8N_BASIC_AUTH_USER=admin
      - N8N_BASIC_AUTH_PASSWORD=supersecret
      - N8N_ENCRYPTION_KEY=PUT_A_LONG_RANDOM_KEY_HERE
      - TZ=Europe/Berlin
    volumes:
      - n8n_data:/home/node/.n8n
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.n8n.rule=Host(`n8n.deinedomain.de`)"
      - "traefik.http.routers.n8n.entrypoints=websecure"
      - "traefik.http.routers.n8n.tls.certresolver=le"
    restart: unless-stopped
  pyapi:
    build: ./python-svc
    expose: [ "8000" ]
    restart: unless-stopped
volumes:
  traefik_letsencrypt:
  n8n_data:
EOF
```
Start:
```
docker compose -f docker-compose-traefik.yml up -d --build
```

5) Datenmodell (SQL DDL für Postgres)

```
-- genierte Tracks
CREATE TABLE gen_tracks (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  created_at TIMESTAMPTZ DEFAULT now(),
  bucket TEXT NOT NULL,
  prompt_id TEXT NOT NULL,
  prompt_text TEXT NOT NULL,
  bpm INT,
  url_wav TEXT,
  url_stems_zip TEXT,
  lufs REAL,
  score_auto REAL
);
-- einzelne Stem-Dateien
CREATE TABLE stems_files (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  gen_track_id UUID REFERENCES gen_tracks(id) ON DELETE CASCADE,
  stem_name TEXT NOT NULL,     -- drums|bass|synth|fx|vocals
  file_url TEXT,
  duration_s REAL,
  rms REAL
);
-- Bewertungen
CREATE TABLE ratings (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  gen_track_id UUID REFERENCES gen_tracks(id) ON DELETE CASCADE,
  "user" TEXT DEFAULT 'sid',
  rating INT CHECK (rating IN (-1,0,1)),
  note TEXT,
  created_at TIMESTAMPTZ DEFAULT now()
);
-- Stem-Auswahl (dein Wunsch je Track)
CREATE TABLE stems_selection (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  gen_track_id UUID REFERENCES gen_tracks(id) ON DELETE CASCADE,
  keep_drums BOOLEAN DEFAULT true,
  keep_bass  BOOLEAN DEFAULT true,
  keep_synth BOOLEAN DEFAULT true,
  keep_fx    BOOLEAN DEFAULT true,
  keep_vocals BOOLEAN DEFAULT false,
  created_at TIMESTAMPTZ DEFAULT now()
);
-- Prompt-Bank (Lernen/Weights)
CREATE TABLE prompt_bank (
  id TEXT PRIMARY KEY,
  bucket TEXT NOT NULL,
  prompt_text TEXT NOT NULL,
  weight REAL DEFAULT 1.0,
  wins INT DEFAULT 0,
  losses INT DEFAULT 0,
```

```
  last_score REAL DEFAULT 0.0,
  last_used_at TIMESTAMPTZ
);
```

> Alternativ: erst Google Sheets, dann Umzug auf Postgres.

6) n8n Workflows (Import■Vorlagen)

6.1 Hello World (jede Minute)


```
{
  "name": "Hello World Every Minute",
  "nodes": [
    {"parameters":{"triggerTimes":{"item":[{"mode":"everyMinute"}]}},"id":"Cron","name":"Cron","type":"n8n-nodes-base.cron","typeVersion":1,"position":[-380,-40]},
    {"parameters":{"functionCode":"return [{json:{message:"Hello World from n8n ■", time:new Date().toISOString()}}];"},"id":"Fn","name":"Function","type":"n8n-nodes-base.function","typeVersion":2,"position":[-160,-40]}
  ],
  "connections":{"Cron":{"main":[[{"node":"Function","type":"main","index":0}]]}}
}
```


6.2 Python■Bridge (Beatport Top → JSON)


```
{
  "name": "Beatport Top via pyapi",
  "nodes": [
    {"parameters":{},"id":"Manual","name":"Manual Trigger","type":"n8n-nodes-base.manualTrigger","typeVersion":1,"position":[-520,-40]},
    {"parameters":{"url":"http://pyapi:8000/beatport_top?genre=hard-techno","responseFormat":"json"},"id":"HTTP","name":"HTTP pyapi","type":"n8n-nodes-base.httpRequest","typeVersion":4,"position":[-300,-40]},
    {"parameters":{"functionCode":"const items=$json.items||[]; return items.map(i=>({json:i}));"},"id":"Flat","name":"Function Flatten","type":"n8n-nodes-base.function","typeVersion":2,"position":[-80,-40]}
  ],
  "connections":{"Manual Trigger":{"main":[[{"node":"HTTP pyapi","type":"main","index":0}]]},"HTTP pyapi":{"main":[[{"node":"Function Flatten","type":"main","index":0}]]}}
}
```


6.3 Rating■Webhook (fallback ohne Telegram)

- **Webhook** (URL: `/rate`) → akzeptiert `gen_track_id`, `rating`
- **HTTP → pyapi /update_prompt_weight** (POST)

Pseudokonfiguration:


```
{
  "name": "Rating Webhook",
  "nodes": [
    {"parameters":{"path":"rate","responseMode":"onReceived","options":{"responseData":"{"ok":true}"}},"id":"WH","name":"Webhook","type":"n8n-nodes-base.webhook","typeVersion":1,"position":[-540,120]},
    {"parameters":{"url":"http://pyapi:8000/update_prompt_weight","options":{"bodyContentType":"json"},"jsonParameters":true,"parameters":{"body":{"prompt_id":"={{$json.gen_track_id}}","rating":"={{$json.rating}}"},"options":{}}},"id":"HTTP","name":"HTTP Update Weight","type":"n8n-nodes-base.httpRequest","typeVersion":4,"position":[-320,120]}
  ],
  "connections":{"Webhook":{"main":[[{"node":"HTTP Update Weight","type":"main","index":0}]]}}
}
```

7) Generierung & Limits

- **Daily cap**: Max 10–20 Jobs
- **Rate limit**: 1 Job / 2–3 Min
- **Queue■Mode** (optional, Produktion): Redis + Worker
  environment:
    - EXECUTIONS_MODE=queue
    - QUEUE_BULL_REDIS_HOST=redis

8) Stems & Audio

- **FFmpeg** (VM): `sudo apt install -y ffmpeg`
- **Demucs**:
  sudo apt install -y python3-pip
  pip install demucs
  demucs --two-stems=vocals input.wav -o /output
- **Mixdown ohne bestimmte Stems** (Beispiel mit ffmpeg amix):
  ffmpeg -i drums.wav -i bass.wav -i synth.wav -filter_complex amix=inputs=3:normalize=0 -c:a pcm_s16le mixed.wav

9) Sicherheit & Betrieb

- NSG: Port 22 nur von deiner IP; 80/443 offen (Traefik) oder nur SSH■Tunnel
- n8n Basic Auth aktiv (ENV)
- Backups: `n8n_data` Volume sichern
- Updates:
  docker compose pull
  docker compose up -d
- Logs: `docker logs n8n -f`

10) Roadmap

- Postgres + Redis (Queue)
- Telegram■Rating■Flow (Inline■Buttons)
- Suno■Integration (Platzhalter■Nodes → echte API)
- Ableton■OSC / LANDR für Mastering
- Trend■Scoring aus Presse/YouTube/Foren

Appendix A — SSH■Alias (komfortabel)

`~/.ssh/config`:

```
Host n8n-vm
  HostName
  User azureuser
  IdentityFile ~/.ssh/vm-n8n_key.pem
  LocalForward 5678 localhost:5678
```

Dann: `ssh n8n-vm` → Tunnel + Login in einem Schritt.

Appendix B — Troubleshooting

- `Permission denied (publickey)`: Pfad/Dateirechte des Keys prüfen (`chmod 600`).
- n8n nicht erreichbar: Tunnel offen? `ssh -L ...` läuft? NSG■Regeln checken.
- Compose baut nicht: `docker compose build --no-cache`.