Ops-Runbook (Kurz & präzise)

Ziel: Stabil starten/stoppen, schnell prüfen, Last konservativ fahren, sauber messen.

1) Start (stabile Baseline)

```
cd "/Users/marisalanger/trancelate-onprem"
ANNI_DEVICE=cpu
ANNI_PREWARM=off
ANNI_GUARD_BATCH=off
MT_WORKERS=1
ANNI_MAX_CONCURRENCY=1
ANNI_TORCH_THREADS=1
ANNI_TORCH_INTEROP=1
ANNI_API_KEY=topsecret
./start_local.sh
```

2) Health & Meta

```
curl -s http://127.0.0.1:8090/health
curl -s http://127.0.0.1:8091/meta
```

3) Prewarm (über Guard)

```
curl -s -H 'Content-Type: application/json' -H 'X-API-Key: topsecret'
  -d '{"pairs":[["de","en"],["fr","en"],["es","en"],["it","en"],["pt","en"],
["nl","en"]]}'
  http://127.0.0.1:8091/prewarm
```

4) Smoke-Tests

```
# A: Plain
curl -s -H 'Content-Type: application/json' -H 'X-API-Key: topsecret'
  -d '{"source":"de","target":"en","text":"Guten Morgen"}'
  http://127.0.0.1:8091/translate

# B: PH+HTML+NUM
curl -s -H 'Content-Type: application/json' -H 'X-API-Key: topsecret'
  -d '{"source":"de","target":"en","text":"Nur heute: {{COUNT}} Plätze frei
bei <strong>{app}</strong> – 2 Tage gültig!"}'
  http://127.0.0.1:8091/translate
```

5) Metriken (Prometheus-Stil)

```
curl -s http://127.0.0.1:8091/metrics
```

Wichtigste Kennzahlen: `anni_uptime_seconds`, `anni_requests_total{route=…}`, `anni_errors_total{route=…}`, `anni_translate_latency_seconds_avg`.

6) Lasttest (konservativ)

```
python3 - <<'PY'
import json, time, statistics, urllib.request, urllib.error,
concurrent.futures
URL="http://127.0.0.1:8091/translate"
H={"Content-Type":"application/json","Accept":"application/json","X-API-
Key":"topsecret"}
TXT="Nur heute: {{COUNT}} Plätze frei bei <strong>{app}</strong> – 2 Tage
gültig!"
PAIRS=[("de","en"),("fr","en"),("es","en"),("it","en"),("pt","en"),
("nl","en")]

def one(s,t):

req=urllib.request.Request(URL,method="POST",headers=H,data=json.dumps({"source":s,"target":t,
    t0=time.perf_counter()
    try:
        with urllib.request.urlopen(req,timeout=30) as r: r.read(); code=200
    except urllib.error.HTTPError as e: code=e.code
    return (time.perf_counter()-t0)*1000.0, code

# Warm
for s,t in PAIRS:
    for _ in range(5): one(s,t)

N,W=30,4
times=[]; fails=0
with concurrent.futures.ThreadPoolExecutor(max_workers=W) as ex:
    futs=[ex.submit(one,s,t) for s,t in PAIRS for _ in range(N)]
    for f in concurrent.futures.as_completed(futs):
        dt,c=f.result(); times.append(dt); fails+=(c!=200)

def pct(a,p):
    a=sorted(a); k=(len(a)-1)*p/100.0; f=int(k); c=min(f+1,len(a)-1)
    return a[f] if f==c else a[f]*(1-(k-f))+a[c]*(k-f)
print(f"sent={len(times)} ok={len(times)-fails} fail={fails}")
print("ms p50=%.1f p90=%.1f p95=%.1f p99=%.1f avg=%.1f" % (
    pct(times,50), pct(times,90), pct(times,95), pct(times,99),
```

```
statistics.mean(times)))
PY
```

Ziel (M2/16GB, CPU): fail≈0, p95 ≈ 1.3–2.0 s.

7) Stop/Reset (hart, falls nötig)

```
lsof -tiTCP:8090,8091 -sTCP:LISTEN | xargs -I{} kill -9 {} 2>/dev/null ||
true
```

8) Tuning-Hebel (ein Regler pro Schritt)

- **MT_WORKERS**: 1 (stabil) → 2 (mehr TPS; Risiko Tail)
- **ANNI_MAX_CONCURRENCY**: 1 → 2 (pro Worker)
- **ANNI_DEVICE**: cpu → mps/cuda (nur mit 1 Worker testen)
- **ANNI_GUARD_BATCH**: off → on (nur wenn Worker stabil; geringer Gewinn)

9) Troubleshooting (symptom → Ursache → Maßnahme)

- **500/502-Spitzen** → paralleler Pipeline-Bau/Überlast → *Worker=1*, `ANNI_MAX_CONCURRENCY=1`, Prewarm fahren.
- **Punkt-Salat/Tag-Duplikate** → Mikro-Segmentierung → Guard auf satzweise Segmente belassen; Dedupe aktiv.
- **zsh meckert „# …“** → Kommentare in Inline-Ketten vermeiden.

TranceLation Engine – Schritt-für-Schritt-Skizze

Überblick

Client → **Guard/Anni (8091)** → **Worker/MT (8090)** → Guard-Post-Edit → Response (Checks/Metriken).

1) Eingang & Auth

- Endpoint: `POST /translate` (Guard)
- Auth: `X-API-Key` (env: `ANNI_API_KEY`)
- Payload: `{source, target, text}`

2) Normalisierung & Erkennung

- Sprachkodes normalisieren (`de`, `en`, …)
- „Invarianten“ erkennen:
  PH `{{…}}`, **APP** `{app}`, **HTML** `<tag>`, **NUM** `1,299.00`, Ranges `2–4`, Versions `HTML5`, Zeit `4 pm`.

3) Freeze der Invarianten

- Ersetzen durch interne Marker (⟦PHk⟧, Zahlen, Tags bleiben 1:1)
- **Never-Translate Glossar** (optional aus `glossary.json`)

4) Satzbasierte Segmentierung

- Robustes Split an Satzende (`.?!`) mit Abkürzungs-Maskierung.

5) MT-Segmente extrahieren

- Aus jedem Satz nur **Textteile** an MT geben;
  PH/HTML/NUM/Interpunktion werden nicht verändert.

6) Routing zum Worker

- HTTP → Worker `/translate` (Single)
  (Batch-Endpoint `/translate_batch` vorhanden, derzeit aus Stabilitätsgründen deaktiviert)

7) Worker – Pipeline (HF/Opus-MT)

- Modellwahl nach Paar (z. B. `Helsinki-NLP/opus-mt-de-en`)
- **Thread-safe Init**: Lock pro Sprachpaar
- **Inferenz-Semaphor**: `ANNI_MAX_CONCURRENCY` pro Prozess
- **Gerätewahl**: `ANNI_DEVICE=cpu|mps|cuda|auto`

8) Rückweg & Restore

- MT-Outputs zurück in Guard
- Marker tolerantes Restore (⟦PHk⟧, NT, VER, RNG), **Tag-Dedupe**

9) Post-Edit (leicht, deterministisch)

- Spacing um `:` und `-/–`
- EN/PT Minis (z. B. „Only today:“ statt „today’s:“)

10) Quality-Checks

- `ph_ok` – alle Platzhalter vorhanden
- `html_ok` – Tag-Signatur identisch
- `num_ok` – Zahlen unverändert repräsentiert
- `paren_ok` – Klammerlogik
- `len_ratio` – Verhältnis Quelle/Ziel

11) Telemetrie

- `/metrics` (Prometheus-Textformat): uptime, requests/errors je Route, avg-Latency
- `/meta`: Engine-Name, Backend-URL, TM-Stats, Flags

12) Betriebsmodi

- **Baseline (empfohlen):** CPU, Worker=1, Sem=1, Guard-Batch=off
- **Durchsatz (vorsichtig):** Worker=2, Sem=2, Prewarm vor Last
- **GPU/MPS (Lab):** `ANNI_DEVICE=mps/cuda`, Worker=1, Sem=1

13) Erweiterungen (Roadmap)

- **Container/Compose**: Guard+Worker mit Healthchecks & Log-Rotation
- **Cache** im Guard (LRU pro Paar)
- **Weitere Sprachen** via Worker-Mapping
- **Copywriter „Anni-CW“** als separater Pfad nach stabiler MT-Basis
- **Metrik-Histogramme** für p50/p90/p95

Anhang – Env-Referenz

- `ANNI_DEVICE` – Zielgerät `cpu|mps|cuda|auto`
- `MT_WORKERS` – Uvicorn-Worker-Prozesse
- `ANNI_MAX_CONCURRENCY` – gleichzeitige Inferences/Prozess
- `ANNI_TORCH_THREADS`, `ANNI_TORCH_INTEROP` – Torch Threading
- `ANNI_GUARD_BATCH` – on/off
- `ANNI_PREWARM` – on/off
- `ANNI_API_KEY` – Header `X-API-Key`

Ende.