HLS映像検索・サムネイル・再生システム構築手順書 — リバースプロキシ対応による公開境界分離構成 —

以下は、貼付「HLS映像検索・サムネイル・再生システム構築手順書.pdf」を前提に、**"今回の修正（= reverse-proxy 化）で追加・変更した箇所だけ"**を、**手順書内にそのまま貼り付けられる形（ファイル全文＋コマンド全文）**でまとめたものである。 （方針は、貼付「映像検索・配信システム構築手順書.pdf」の **reverse-proxy（Nginx）で API を隠蔽する方式**を踏襲している 。）

---

0. 背景と目的（今回の差分の意義）

背景

従来手順では、FastAPI が **HLS（/hls）やサムネイル（/thumbs）やUI（/ui）を直接配信**する構成になっており 、ブラウザからのアクセス経路が「API 直アクセス」になりがちです。また DB もホストに port 公開して運用することが多い（例：`15432:5432`） 。

目的（今回）

- ブラウザから見える公開点を **http://localhost/（reverse-proxy）に一本化**する
- **API/DB/HLS/サムネイルを"フロントから直接見えない"**（= 直接ポート露出や直配信を避ける）構造へ寄せる
- 方式は「映像検索・配信システム構築手順書.pdf」の reverse-proxy 構成（`location /api/ { proxy_pass ... }` 等）を踏襲

---

1. 今回の修正で追加するディレクトリ

既存 `~/video_hls_project` の直下に、以下を **新規作成**する。

```
cd ~/video_hls_project

mkdir -p api
mkdir -p reverse-proxy/frontend
```

---

2. 今回の修正で「新規に作る（または置き換える）」ファイル一覧

- `api/Dockerfile`（新規）
- `api/app.py`（新規：※従来の `backend_api/app.py` を "コンテナ運用前提" にした版）
- `reverse-proxy/Dockerfile`（新規）
- `reverse-proxy/nginx.conf`（新規）
- `reverse-proxy/frontend/index.html`（新規：UIをreverse-proxy側に同梱）
- `docker-compose.yml`（置き換え）

---

3. ファイル全文：api/Dockerfile（新規）

作成先： `~/video_hls_project/api/Dockerfile`

```
cat << 'EOF' > ~/video_hls_project/api/Dockerfile
FROM python:3.11-slim

WORKDIR /app

RUN apt-get update && apt-get install -y --no-install-recommends \
    build-essential libpq-dev \
 && rm -rf /var/lib/apt/lists/*

RUN pip install --no-cache-dir \
    fastapi uvicorn psycopg2-binary python-multipart

COPY app.py /app/app.py

EXPOSE 8000
CMD ["uvicorn", "app:app", "--host", "0.0.0.0", "--port", "8000"]
EOF
```

---

4. ファイル全文：api/app.py（新規）

作成先： `~/video_hls_project/api/app.py`

ポイント：

- DB内 `m3u8_path`（例：`videos_hls/clip01/index.m3u8`）を、返却URL `/hls/clip01/index.m3u8` に変換するロジックは従来PDFの考え方を踏襲している。
- `/hls` `/thumbs` を FastAPI で静的マウントする方式自体は従来PDF同様
- ただし公開は reverse-proxy 経由に統一する（後述 nginx.conf）

```
cat << 'EOF' > ~/video_hls_project/api/app.py
from fastapi import FastAPI, HTTPException
from fastapi.staticfiles import StaticFiles
from typing import Optional
import psycopg2
import psycopg2.extras
import os
from pathlib import Path

# ========= パス設定（コンテナ内） =========
# docker-compose で /data にホストの video_hls_project をマウントする前提
BASE_DIR = Path("/data")

HLS_DIR   = BASE_DIR / "videos_hls"
THUMB_DIR = BASE_DIR / "thumbnails"
```

```
# ========= DB 接続設定（コンテナ内から db サービスへ） =========
DB_HOST = os.getenv("DB_HOST", "db")
DB_PORT = int(os.getenv("DB_PORT", "5432"))
DB_NAME = os.getenv("DB_NAME", "video_db")
DB_USER = os.getenv("DB_USER", "video_user")
DB_PASS = os.getenv("DB_PASS", "password")

def get_conn():
    return psycopg2.connect(
        host=DB_HOST,
        port=DB_PORT,
        dbname=DB_NAME,
        user=DB_USER,
        password=DB_PASS,
    )

app = FastAPI()

@app.get("/health")
def health():
    return {"status": "ok"}

# ========= 静的ファイル =========
# （従来手順では FastAPI が直配信していたが、今回は reverse-proxy 経由でのみ公開する）
app.mount("/hls", StaticFiles(directory=HLS_DIR), name="hls")
app.mount("/thumbs", StaticFiles(directory=THUMB_DIR), name="thumbs")

def to_hls_url(m3u8_path: str) -> str:
    """
    DBの m3u8_path (例: videos_hls/clip01/index.m3u8)
    を、ブラウザ向け URL (/hls/clip01/index.m3u8) に変換する。
    （従来手順書の変換仕様を踏襲）
    """
    prefix = "videos_hls/"
    if m3u8_path.startswith(prefix):
        rel = m3u8_path[len(prefix):]   # "clip01/index.m3u8"
    else:
        rel = m3u8_path
    return f"/hls/{rel}"

def to_thumb_url(chunk_id: str) -> str:
    """
    チャンク代表サムネイルは 00000.jpg を返す（従来通り）。
    """
    return f"/thumbs/{chunk_id}/00000.jpg"

@app.get("/api/chunks")
def list_chunks(tag: Optional[str] = None):
    """
    映像一覧API
    - /api/chunks
    - /api/chunks?tag=test
    """
    conn = get_conn()
```

```
    try:
        cur = conn.cursor(cursor_factory=psycopg2.extras.RealDictCursor)
        if tag:
            sql = """
            SELECT camera_id, chunk_id, title, m3u8_path, duration_sec, tags,
created_at
            FROM train_camera_chunk
            WHERE tags @> ARRAY[%s]::text[]
            ORDER BY id;
            """
            cur.execute(sql, (tag,))
        else:
            sql = """
            SELECT camera_id, chunk_id, title, m3u8_path, duration_sec, tags,
created_at
            FROM train_camera_chunk
            ORDER BY id;
            """
            cur.execute(sql)

        rows = cur.fetchall()
        result = []
        for row in rows:
            d = dict(row)
            d["hls_url"] = to_hls_url(d["m3u8_path"])
            d["thumb_url"] = to_thumb_url(d["chunk_id"])
            result.append(d)
        return result
    finally:
        conn.close()

@app.get("/api/chunks/{chunk_id}")
def get_chunk(chunk_id: str):
    """
    chunk_id（clip01 等）で1件取得
    """
    conn = get_conn()
    try:
        cur = conn.cursor(cursor_factory=psycopg2.extras.RealDictCursor)
        sql = """
        SELECT camera_id, chunk_id, title, m3u8_path, duration_sec, tags,
created_at
        FROM train_camera_chunk
        WHERE chunk_id = %s;
        """
        cur.execute(sql, (chunk_id,))
        row = cur.fetchone()
        if not row:
            raise HTTPException(status_code=404, detail="chunk not found")

        d = dict(row)
        d["hls_url"] = to_hls_url(d["m3u8_path"])
        d["thumb_url"] = to_thumb_url(d["chunk_id"])
        return d
```

```
    finally:
        conn.close()

@app.get("/api/chunks/{chunk_id}/segments")
def get_segments(chunk_id: str):
    """
    指定 chunk の index.m3u8 を読み、TSセグメント情報を返す。
    （従来手順書にある /api/chunks/{chunk_id}/segments の考え方を踏襲）
    """
    m3u8_path = HLS_DIR / chunk_id / "index.m3u8"
    if not m3u8_path.exists():
        raise HTTPException(status_code=404, detail="m3u8 not found")

    segments = []
    current_start = 0.0

    with m3u8_path.open("r", encoding="utf-8") as f:
        lines = [line.strip() for line in f if line.strip()]

    i = 0
    idx = 0
    while i < len(lines):
        line = lines[i]
        if line.startswith("#EXTINF:"):
            dur_part = line.split(":", 1)[1]
            dur_str = dur_part.split(",", 1)[0]
            try:
                dur = float(dur_str)
            except ValueError:
                dur = 0.0

            if i + 1 >= len(lines):
                break

            ts_name = lines[i + 1]  # "00000.ts" 等

            segments.append({
                "index": idx,
                "ts": ts_name,
                "start_sec": current_start,
                "duration_sec": dur,
                "thumb_url": f"/thumbs/{chunk_id}/{ts_name.replace('.ts',
'.jpg')}",
            })

            current_start += dur
            idx += 1
            i += 2
        else:
            i += 1

    return segments
EOF
```

5. ファイル全文：reverse-proxy/Dockerfile（新規）

作成先： `~/video_hls_project/reverse-proxy/Dockerfile`

（「映像検索・配信システム構築手順書.pdf」の Nginx コンテナ作り方を踏襲 ）

```
cat << 'EOF' > ~/video_hls_project/reverse-proxy/Dockerfile
FROM nginx:alpine
COPY nginx.conf /etc/nginx/nginx.conf
COPY frontend /usr/share/nginx/html
EOF
```

6. ファイル全文：reverse-proxy/nginx.conf（新規）

作成先： `~/video_hls_project/reverse-proxy/nginx.conf`

ポイント：

- `/api/` を `api:8000` に reverse-proxy（方式は踏襲）
- **`/hls/` と `/thumbs/` も api 側へ proxy**（= ブラウザは reverse-proxy 以外に触れない）

```
cat << 'EOF' > ~/video_hls_project/reverse-proxy/nginx.conf
events {}

http {
  server {
    listen 80;

    # ---- UI（静的）----
    location / {
      root /usr/share/nginx/html;
      index index.html;
      try_files $uri $uri/ /index.html;
    }

    # ---- API ----
    location /api/ {
      proxy_pass http://api:8000/;
      proxy_set_header Host $host;
      proxy_set_header X-Real-IP $remote_addr;
      proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }

    # ---- HLS（m3u8/ts）----
    location /hls/ {
      proxy_pass http://api:8000/hls/;
      proxy_set_header Host $host;
```

```
      proxy_set_header X-Real-IP $remote_addr;
      proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }

    # ---- Thumbnails (jpg) ----
    location /thumbs/ {
      proxy_pass http://api:8000/thumbs/;
      proxy_set_header Host $host;
      proxy_set_header X-Real-IP $remote_addr;
      proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }
  }
}
EOF
```

---

7. ファイル全文：reverse-proxy/frontend/index.html（新規）

作成先： `~/video_hls_project/reverse-proxy/frontend/index.html`

※UIは reverse-proxy 配下で配信し、API は `/api/...` を叩く（踏襲の方向性：reverse-proxyで公開点を分離）。

```
cat << 'EOF' > ~/video_hls_project/reverse-proxy/frontend/index.html
<!doctype html>
<html lang="ja">
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width,initial-scale=1" />
  <title>HLS映像検索・サムネイル・再生</title>
  <style>
    body { font-family: sans-serif; margin: 16px; }
    .row { display: flex; gap: 16px; }
    .col { flex: 1; min-width: 320px; }
    .grid { display: grid; grid-template-columns: repeat(auto-fill, minmax(160px,
1fr)); gap: 10px; }
    .card { border: 1px solid #ddd; padding: 8px; cursor: pointer; }
    .card:hover { background: #fafafa; }
    img { max-width: 100%; height: auto; display: block; }
    video { width: 100%; background: #000; }
    .muted { color: #666; font-size: 12px; }
  </style>
</head>
<body>
  <h1>HLS映像検索・サムネイル・再生</h1>

  <div style="margin: 8px 0;">
    <label>タグ: <input id="tagInput" placeholder="例: test" /></label>
    <button id="searchBtn">検索</button>
    <span id="status" class="muted"></span>
  </div>
```

```
<div class="row">
  <div class="col">
    <h2>■ サムネイル表示部（映像単位）</h2>
    <div id="chunkGrid" class="grid"></div>
  </div>

  <div class="col">
    <h2>■ セグメントサムネイル表示部（選択中の映像）</h2>
    <div id="segmentGrid" class="grid"></div>

    <h2 style="margin-top:16px;">■ 再生</h2>
    <video id="video" controls playsinline></video>
    <div id="nowPlaying" class="muted"></div>
  </div>
</div>

<script src="https://cdn.jsdelivr.net/npm/hls.js@latest"></script>
<script>
  const statusEl = document.getElementById('status');
  const chunkGrid = document.getElementById('chunkGrid');
  const segmentGrid = document.getElementById('segmentGrid');
  const videoEl = document.getElementById('video');
  const nowPlayingEl = document.getElementById('nowPlaying');

  let hls = null;

  function setStatus(msg) { statusEl.textContent = msg || ''; }

  async function fetchJSON(url) {
    const res = await fetch(url);
    if (!res.ok) throw new Error(`${res.status} ${res.statusText}`);
    return await res.json();
  }

  function playHLS(url) {
    if (hls) { hls.destroy(); hls = null; }
    if (videoEl.canPlayType('application/vnd.apple.mpegurl')) {
      videoEl.src = url;
    } else if (window.Hls && Hls.isSupported()) {
      hls = new Hls();
      hls.loadSource(url);
      hls.attachMedia(videoEl);
    } else {
      alert('このブラウザはHLS再生に対応していません（hls.jsが使えません）。');
    }
  }

  function clearSegments() {
    segmentGrid.innerHTML = '';
  }

  function renderChunks(chunks) {
    chunkGrid.innerHTML = '';
```

```
      chunks.forEach(c => {
        const div = document.createElement('div');
        div.className = 'card';
        div.innerHTML = `
          <img src="${c.thumb_url}" alt="">
          <div><b>${c.title || c.chunk_id}</b></div>
          <div class="muted">${c.chunk_id} / ${(c.tags||[]).join(', ')}</div>
        `;
        div.onclick = async () => {
          setStatus(`segments取得中: ${c.chunk_id}`);
          nowPlayingEl.textContent = `選択中: ${c.chunk_id}`;
          clearSegments();

          // セグメント一覧
          const segs = await
fetchJSON(`/api/chunks/${encodeURIComponent(c.chunk_id)}/segments`);
          renderSegments(c.chunk_id, segs);

          // 再生
          playHLS(c.hls_url);
          setStatus('');
        };
        chunkGrid.appendChild(div);
      });
    }

    function renderSegments(chunkId, segs) {
      segmentGrid.innerHTML = '';
      segs.forEach(s => {
        const div = document.createElement('div');
        div.className = 'card';
        const thumb = s.thumb_url;
        div.innerHTML = `
          <img src="${thumb}" alt="">
          <div class="muted">#${s.index} ${s.ts}</div>
          <div class="muted">${s.start_sec.toFixed(2)}s
(+${s.duration_sec.toFixed(2)}s)</div>
        `;
        segmentGrid.appendChild(div);
      });
    }

    async function loadChunks() {
      const tag = document.getElementById('tagInput').value.trim();
      const url = tag ? `/api/chunks?tag=${encodeURIComponent(tag)}` :
'/api/chunks';
      setStatus('chunks取得中...');
      const chunks = await fetchJSON(url);
      renderChunks(chunks);
      clearSegments();
      nowPlayingEl.textContent = '';
      setStatus(`件数: ${chunks.length}`);
    }
```

```
    document.getElementById('searchBtn').onclick = loadChunks;

    // 初期ロード
    loadChunks().catch(e => {
      console.error(e);
      setStatus('初期ロード失敗: ' + e.message);
    });
  </script>
</body>
</html>
EOF
```

---

8. ファイル全文：docker-compose.yml（置き換え）

作成先： `~/video_hls_project/docker-compose.yml`

ポイント：

- db / api / reverse-proxy の 3サービス化（reverse-proxy の ports 80:80 のみ公開）
- db の ports 公開は **原則やめる**（必要ならコメント解除で運用）
- api は `~/video_hls_project` を `/data` にマウントして HLS/サムネイルを参照する

（reverse-proxy + api + db の並びは「映像検索・配信システム構築手順書.pdf」の構成を踏襲 ）

```
cat << 'EOF' > ~/video_hls_project/docker-compose.yml
version: "3.9"

services:
  db:
    image: postgis/postgis:16-3.4
    container_name: train-video-db
    environment:
      POSTGRES_DB: video_db
      POSTGRES_USER: video_user
      POSTGRES_PASSWORD: password

    # 重要：原則、ホストへ公開しない（フロントから見えない）
    # 必要ならデバッグ時だけ一時的に開ける：
    # ports:
    #   - "15432:5432"

    volumes:
      - ./db_init:/docker-entrypoint-initdb.d
      - pgdata:/var/lib/postgresql/data

  api:
    build: ./api
    container_name: train-video-api
    environment:
      DB_HOST: db
```

```
      DB_PORT: "5432"
      DB_NAME: video_db
      DB_USER: video_user
      DB_PASS: password
    depends_on:
      - db
    volumes:
      # ホストの ~/video_hls_project を /data として参照（HLS/サムネイルを読む）
      - ./:/data:ro

    # 重要：api もホストへ公開しない（reverse-proxy 経由のみ）
    # ports:
    #   - "18000:8000"

  reverse-proxy:
    build: ./reverse-proxy
    container_name: train-video-frontend
    ports:
      - "80:80"
    depends_on:
      - api

volumes:
  pgdata:
EOF
```

---

9. 起動方法（今回版）

```
cd ~/video_hls_project

docker compose down
docker compose up -d --build

docker ps --format "table {{.Names}}\t{{.Status}}\t{{.Ports}}"
```

期待：`train-video-frontend` だけが `0.0.0.0:80->80/tcp` を持ち、db/api は host port を持たない。

---

10. 確認方法（今回版）

10.1 reverse-proxy 経由で API を確認

```
curl -i http://localhost/api/health
curl -i http://localhost/api/chunks
```

10.2 HLS の疎通確認（例：clip03）

```
curl -I http://localhost/hls/clip03/index.m3u8
```

10.3 ブラウザ確認

- Web UI：`http://localhost/`
- 映像一覧（JSON）：`http://localhost/api/chunks`

---

11. 運用上の注意（今回の「フロントから見えない」化の要点）

- **DBはホストへ port 公開しない**（原則） 従来の `15432:5432` 公開は、必要時のみ一時的に使う扱いに変更（元手順では公開例あり ）。
- **APIもホストへ port 公開しない** ブラウザは `http://localhost/` にしか触れず、`/api` `/hls` `/thumbs` は全て Nginx が中継。