Zero-Cost Local AI: Menjalankan "Tiny LLM" di CPU Lokal

Eksperimen Arsitektur AI Menggunakan FastAPI & OpenClaw — tanpa GPU mahal, tanpa cloud, tanpa biaya. Semua berjalan di laptop standar kamu hari ini.

LAB GUIDE

ADVANCED IT · SYDNEY

Objektif Lab Hari Ini

Satu tujuan utama: **membuktikan bahwa arsitektur AI kelas Enterprise bisa dibangun dan diuji sepenuhnya secara lokal** — tanpa GPU mahal, tanpa koneksi cloud, dan tanpa biaya lisensi apapun.

🧠 Jalankan Model AI

Inference CPU-only menggunakan model mikro GGUF di bawah 1 GB langsung dari laptop kamu.

⚙️ Bangun Backend API

Buat REST API production-ready dengan FastAPI yang terhubung ke LLM lokal via OpenClaw.

🔬 Uji Arsitekturnya

Testing end-to-end via Swagger UI bawaan — persis seperti workflow Backend Developer nyata.

ⓘ Di akhir sesi ini, kamu punya working AI backend yang siap dikembangkan ke skala Enterprise minggu depan.

Konsep Tiny AI & Quantization

Model AI modern seperti LLaMA atau Qwen awalnya berukuran puluhan gigabyte. Teknik **quantization** memadatkan bobot numerik dari 32-bit floating point menjadi 4-bit atau 8-bit integer — mengurangi ukuran hingga 75% tanpa kehilangan akurasi signifikan.

Format GGUF

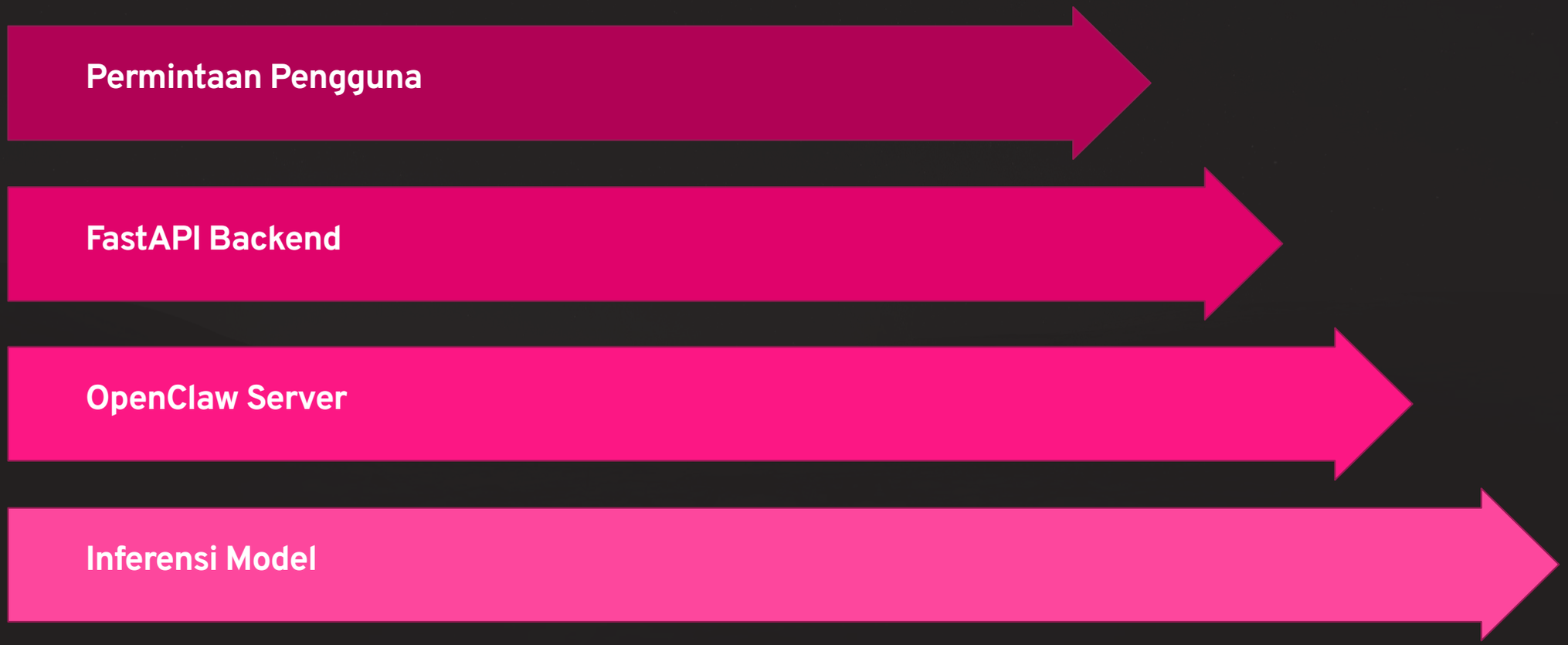
Format file tunggal yang dioptimalkan untuk inferensi CPU. Menggantikan format lama GGML dengan metadata lebih kaya dan kompatibilitas lintas platform.

Mengapa Ini Penting?

- Model 7B parameter → muat dalam RAM 4 GB
- Inferensi berjalan di CPU biasa tanpa CUDA
- Latensi cukup untuk development & prototyping
- Cocok untuk laptop standar mahasiswa

Arsitektur Sistem Hari Ini

Berikut adalah alur lengkap request dari pengguna hingga respons AI — semua berjalan secara lokal di satu mesin, tanpa satu pun paket data keluar ke internet.



Pola arsitektur ini identik dengan deployment Enterprise: hanya bedanya, di production kamu ganti `localhost:11434` dengan endpoint GPU cloud dan model mikro dengan Llama-3 penuh.

Langkah 1: Persiapan Environment

STEP 1 / 5

Buka terminal di laptop kamu. Pastikan Python 3.10+ sudah terinstall. Semua dependensi yang dibutuhkan bisa diinstall dalam satu perintah — estimasi waktu **kurang dari 2 menit** dengan koneksi WiFi kampus.

```
# Install semua dependensi sekaligus
pip install fastapi uvicorn openclaw huggingface_hub

# Verifikasi instalasi berhasil
python -c "import fastapi, uvicorn; print('
```

Langkah 2: Mengunduh Model Mikro

STEP 2 / 5

Kita gunakan **Qwen-1.5-0.5B-GGUF** — model dengan hanya 500 juta parameter yang dikuantisasi ke format Q4_K_M. Ukuran file sekitar **600 MB**, proses download memakan waktu **2–5 menit** di jaringan WiFi standar.

```
# Download model GGUF dari HuggingFace
huggingface-cli download \
  Qwen/Qwen1.5-0.5B-Chat-GGUF \
  qwen1_5-0_5b-chat-q4_k_m.gguf \
  --local-dir ./models/

# Cek file berhasil diunduh (~600MB)
ls -lh ./models/
```

Kenapa Qwen-0.5B? Model ini cukup "cerdas" untuk demo percakapan sederhana, namun ringan sehingga bahkan laptop dengan RAM 8 GB pun mampu menjalankannya tanpa hambatan berarti.

Langkah 3: Menghidupkan OpenClaw Server

STEP 3 / 5

OpenClaw berperan sebagai **inference server lokal** yang menyediakan API kompatibel OpenAI di `localhost:11434`. Backend FastAPI kita nantinya akan menembak endpoint ini — bukan internet.

```
# Jalankan OpenClaw dengan model yang sudah diunduh
openclaw serve \
 --model ./models/qwen1_5-0_5b-chat-q4_k_m.gguf \
 --host 127.0.0.1 \
 --port 11434

# Output yang diharapkan:
#
```

Langkah 4: Membuat "Otak" Backend

STEP 4 / 5

Buat file `main.py` di folder proyek kamu. Kode berikut membuat FastAPI app dengan endpoint `/chat` yang menerima pesan dari user, meneruskannya ke OpenClaw lokal, lalu mengembalikan respons AI.

```
from fastapi import FastAPI
from pydantic import BaseModel
from openclaw import OpenClaw

app = FastAPI(title="Local AI Backend")
client = OpenClaw(base_url="http://127.0.0.1:11434")

class ChatRequest(BaseModel):
    message: str

@app.post("/chat")
async def chat(req: ChatRequest):
    response = client.chat.completions.create(
        model="qwen1_5-0_5b-chat-q4_k_m.gguf",
        messages=[{"role": "user", "content": req.message}]
    )
    return {
        "reply": response.choices[0].message.content,
        "model": "Qwen-0.5B-Local"
    }
```

Interface `client.chat.completions.create()` identik dengan OpenAI SDK — skill ini langsung transferable ke production environment berbasis cloud.

Langkah 5: Testing Endpoint via Swagger UI

STEP 5 / 5

FastAPI menyertakan dokumentasi interaktif Swagger UI secara otomatis — tidak perlu tool tambahan seperti Postman. Ini adalah cara paling cepat untuk memverifikasi API kamu berfungsi end-to-end.

```
# Jalankan server FastAPI (terminal baru)
uvicorn main:app --host 0.0.0.0 --port 8000 --reload

# Output yang diharapkan:
# INFO: Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
# INFO: Application startup complete.
```

1 **Buka Swagger UI**

Arahkan browser ke `http://localhost:8000/docs` — antarmuka testing interaktif akan muncul.

2 **Klik endpoint `POST /chat`**

Pilih "Try it out", masukkan pesan seperti `{"message": "Apa itu quantization?"}`, lalu klik Execute.

3 **Baca Respons AI Lokal**

Server akan merespons dalam beberapa detik dengan jawaban yang di-generate sepenuhnya di CPU laptop kamu.

Kesimpulan & Persiapan Minggu Depan

Selamat — kamu baru saja membangun **AI backend berbasis LLM yang berjalan 100% lokal**, tanpa biaya, tanpa GPU, dan tanpa internet. Ini bukan mainan: arsitektur yang sama digunakan di enterprise production.

Hari Ini ✅

FastAPI + OpenClaw + Qwen-0.5B di CPU lokal. Proof of concept arsitektur AI enterprise.

Minggu Depan 🚀

Ganti model mikro dengan **Llama-3 di Cloud GPU**. Kodenya? Hampir tidak berubah — hanya URL dan nama model.

Key Insight: Logika arsitektur ini identik di semua skala. Skill yang kamu bangun hari ini — FastAPI, async endpoint, LLM client interface — adalah skill yang sama yang dipakai di Fortune 500 tech companies.