สร้าง Local AI ใช้เองในบ้านและองค์กร

คู่มือเริ่มสร้าง AI ส่วนตัว ลดค่า API ปกป้องข้อมูล
ใช้กับงานเอกสาร ช่วย Coding และสร้าง Agent เบื้องต้น

สารบัญ

บทนำ 1

บทที่ 1 — ทำไมคนใช้ Cloud AI ถึงเริ่มสนใจ Local AI 2

- คำตอบที่หัวหน้าอยากได้จริง ๆ — มันขึ้นกับ tier 2
- Local AI คืออะไร แบบสั้นที่สุด 4
- เหตุผลจริง ๆ ที่คนเริ่มมอง Local AI 4
- ก่อนไปต่อ ขออธิบาย 3 ความเข้าใจผิดที่ต้องเคลียร์ 5
- Local ไม่ใช่ตัวแทน Cloud — มันคืออีกเครื่องมือ 6
- Cloud คุมอะไรได้/ไม่ได้ vs Local คุมอะไรได้/ต้องดูแลอะไร 7
- หนังสือเล่มนี้จะพาคุณไปถึงไหน 8
- เลือกเส้นทางอ่านของคุณ — Reader Path Map 9
- คุณควรลอง Local AI ไหม — เช็กตัวเองก่อน 10
  - เช็กลิสต์: คุณควรลอง Local AI ไหม 10
- Key takeaways 11
- ก้าวต่อไป 11

บทที่ 2 — Cloud AI, Local AI และ Hybrid AI ต่างกันอย่างไร

- สามทางเลือก เหมือนเรื่องที่อยู่อาศัย
- Cloud / Local / Hybrid คืออะไรกันแน่
- สิ่งที่ย้ายมาอยู่บนบ่าคุณ เมื่อเลือก Local
- Local AI กับ Local LLM ไม่ใช่คำเดียวกัน
- “Open” ไม่ได้แปลว่าเหมือนกันหมด
- วางสามแบบเทียบกัน ทีละมิติ
- เรื่องที่ต้องเคลียร์ — “local = ปลอดภัยอัตโนมัติ” ไม่จริง
- Hybrid ไม่ใช่ทางลัด — มันคือการเลือกที่ฉลาด
- สถานการณ์แบบนี้ เอียงไปทางไหน
- ทดสอบกับงานจริงของคุณ
- Key takeaways
- ก้าวต่อไป

บทที่ 3 งานแบบไหนควรย้ายมา Local และงานไหนควรอยู่บน Cloud

- ทำไม “Local หรือ Cloud” ถึงเป็นคำถามที่ตอบยาก
- เริ่มจากคำถามเดียว ข้อมูลนี้ระดับไหน
  - เรื่อง PDPA ที่ผู้อ่านไทยต้องไม่มองข้าม
- เกณฑ์เลือก 5 ด้าน หลังจากรู้ระดับข้อมูลแล้ว
- แผนที่งาน เอียงไปทางไหน
- ต้นทุนที่แท้จริงของ Local ที่หลายคนลืมคิด
- Hybrid คือจุดลงตัวของหลายคน
- ลงมือจริง กรอกงาน 3 อย่างของคุณ
  - Worksheet: Local AI Use Case Selector
  - Cloud vs Local Decision Matrix — สูตรอ่านผล
- ข้อผิดพลาดที่พบบ่อยและข้อควรระวัง
  - Checklist: กับดักที่ต้องเลี่ยงเวลาตัดสินใจ Local vs Cloud
- Key takeaways
- ก้าวต่อไป

บทที่ 4 Hardware, Model และต้นทุนที่ต้องเข้าใจก่อนเริ่ม

- ทำไมการอ่าน spec ไม่ออกถึงทำให้เจ็บตัว
- ขนาดโมเดล (7B, 70B) บอกอะไร
- Quantization (Q4, Q8) คือการบีบไฟล์
- VRAM กับ RAM คือ “ขนาดโต๊ะทำงาน”
- GPU, CPU และ Apple Silicon ต่างกันตรงความเร็ว
- Context window กินหน่วยความจำเพิ่มอย่างไร
- ตารางสรุป เครื่องแบบนี้รันโมเดลแบบไหนไหว
  - Hardware Requirement Cheat Sheet
- เริ่มต้นโดยไม่ต้องซื้อของใหม่
- แล้วเรื่องต้นทุนล่ะ คิดยังไงดี
- เครื่องมือลงมือ ประเมินเครื่องและเลือกโมเดล
  - Hardware Planning Checklist
  - Model Selection Worksheet
  - ทดสอบกับงานจริงของคุณ
- ข้อผิดพลาดที่พบบ่อยและข้อควรระวัง
- Key takeaways
- ก้าวต่อไป

บทที่ 5 Tool Stack สำหรับ Local AI: Ollama, LM Studio, Open WebUI, llama.cpp และ API

- ทุก tool ทำสิ่งเดียวกันเบื้องหลัง
- สามชั้นที่ทำให้เลิกสับสน
- รู้จัก tool ทีละตัว แบบสั้นแต่พอตัดสินใจ
- GGUF: ไฟล์มาตรฐานที่ทำให้ย้าย tool ได้
- “รันได้” ไม่เท่ากับ “รันดี”
- เลือก tool ของคุณใน 5 นาที
  - ตารางเปรียบเทียบประเภทเครื่องมือ
  - Decision tree: เลือกตามเป้าหมาย
- Mini Project 1: ติดตั้ง Local AI และเปิดแชตตัวแรก
  - Local AI Installation Checklist
  - ทดสอบกับงานจริงของคุณ
- ข้อผิดพลาดที่พบบ่อยและข้อควรระวัง
- Key takeaways
- ก้าวต่อไป

บทที่ 6 สร้าง Local AI Chat ใช้เองแบบง่าย

- หน้าตาเหมือนเดิม แต่แหล่งคำตอบย้ายมาในบ้านคุณ
- System prompt คือการบรีฟงานให้ผู้ช่วยก่อนเริ่มทุกครั้ง
  - ตั้ง system prompt ได้ตรงไหน
  - ก่อนตั้งกับหลังตั้ง ต่างกันแค่ไหน
- Context window คือกระดานความจำที่มีขอบเขต
  - สิ่งที่เกิดขึ้นเมื่อกระดานเต็ม และทำไมมันร้ายกว่าที่ฟัง
  - สี่สัญญาณว่า context กำลังมีปัญหา
  - วิธีจัดการแบบใช้ได้จริง
- Chat loop รอบเดียวที่เกิดทุกครั้งที่คุณพิมพ์
- Local กับ Cloud ต่างกันตรงไหนในชีวิตจริง
- ข้อจำกัดที่ต้องรู้ก่อน โมเดลเล็กตอบมั่วได้แบบมั่นใจ
- ชุด system prompt ที่ก็อปไปใช้ได้ทันที
  - ทดสอบกับงานจริงของคุณ
  - Checklist ก่อนใช้ Local Chat ทำงานจริง
- Key takeaways
- ก้าวต่อไป

บทที่ 7 เรียก Local AI ผ่าน Python และ API

API คืออะไร เมื่ออธิบายเป็นภาษาคน

API ในบทนี้มีสองความหมาย แยกให้ชัดก่อน

ข่าวดีสำหรับคนเคยเขียนโค้ดเรียก AI มาก่อน

ของเดิมต่อได้ แต่ไม่ใช่ "เหมือน 100%"

ภาพรวม: คำขอเดินทางยังไงในเครื่องคุณ

เขียนโค้ด Python เรียก Local AI

ทำให้คำตอบ "ไหล" ออกมาทีละคำด้วย streaming

Cloud API กับ Local API ต่างกันตรงไหนในทางปฏิบัติ

ถ้าใช้ LM Studio แทน Ollama

เรื่อง security ที่ต้องรู้ก่อนเปิดให้คนอื่นใช้

API Testing Checklist

Mini Project 2: เรียก Local AI ผ่าน Python API

ทดสอบกับงานจริงของคุณ

Key takeaways

ก้าวต่อไป

บทที่ 8 Local Document AI ใช้ AI กับไฟล์และเอกสารภายใน

งานที่ทำได้ทันทีคือสรุปไฟล์เดียว

Mini Project 3: สรุป meeting-notes.txt

ทำไมถามตอบหลายเอกสารต้องมี RAG

pipeline ของ RAG มีกี่ขั้น

embedding model และ vector store คืออะไร และทำไมต้อง local ทั้งคู่

เอกสารภาษาไทยต้องเลือก embedding model ให้ถูก

งานเอกสารแบบไหนที่ Local AI ช่วยได้บ้าง

Mini Project 4: Document Q&A จากโฟลเดอร์ docs/

RAG ลด hallucination แต่ไม่กำจัด — ทำไมต้องตรวจคำตอบเสมอ

Document AI Workflow Checklist

ทดสอบกับงานจริงของคุณ (เน้นเอกสารภาษาไทย)

Key takeaways

ก้าวต่อไป

บทที่ 9 Local Coding Assistant ใช้ AI ช่วยอ่านโค้ด แก้ error และสร้าง test

- ทำไมต้องใช้ Local AI กับงานโค้ด
- ห้างานที่ Local AI ช่วยงานโค้ดได้จริง
- ต่อ Local AI เข้ากับ editor ที่คุณใช้อยู่
- เลือกโมเดลสำหรับงานโค้ดยังไง
- ความจริงที่ต้องพูดตรงๆ Local ยังไม่เท่า Cloud ในงานโค้ดยาก
- Coding Assistant Prompt Pack
  - Mini Project 5: ใช้ Local AI อธิบาย error และช่วย refactor
- ทำไมต้องตรวจผลทุกครั้งก่อน merge
  - ทดสอบกับงานจริงของคุณ
- Key takeaways
- ก้าวต่อไป

บทที่ 10 Local AI Agent เบื้องต้น Tools, API, Connectors และ MCP

- แยกห้าคำที่ปนกันให้ออกก่อน
- Agent ต่างจากแชทบอทที่เราคุ้นยังไง
- แล้ว MCP มันคืออะไร และทำไมคนพูดถึงกันเยอะ
- ทำไม Mini Project ของเรา “ยังไม่ใช่” MCP
- วงจรการทำงานของ agent หน้าตาเป็นยังไง
- ทำไมแค่ “สั่งห้าม” ถึงไม่พอ — เรื่องจริงที่ต้องรู้
  - Checklist ความปลอดภัยของ Agent
- Mini Project 6: Local Agent อ่านไฟล์จากโฟลเดอร์ที่อนุญาต
- Mini Project 7: Agent เรียก API ภายใน (จำลอง) แล้วสรุปข้อมูล
  - ทดสอบกับงานจริงของคุณ
- Key takeaways
- ก้าวต่อไป

บทที่ 11 ใช้ Local AI ในองค์กร Privacy, Security, Governance และ Architecture

- Local ปลอดภัยกว่า Cloud จริงไหม ผมขอวางความคาดหวังให้ตรงก่อน
- คำถามแรกก่อนขยายออกไป ข้อมูลที่ทีมจะใส่ AI มันชั้นไหน
- Security พื้นฐาน เหมือนเปิดร้านให้คนเข้า
- Privacy และ PDPA สำหรับองค์กรไทย
- License ของโมเดล ดาวน์โหลดได้ ไม่ได้แปลว่าใช้เชิงพาณิชย์ได้
- Governance สำหรับทีมเล็ก ไม่ต้องซับซ้อนแต่ต้องมี
- Architecture ระดับแนวคิด บ้าน กับ ทีมเล็ก
- ข้อผิดพลาดที่พบบ่อย
- ทดสอบกับงานจริงของคุณ
- Data Privacy Checklist
- Checklist ก่อนปล่อยให้ทีมใช้ (Go / No-Go)
- ไม่มีปลอดภัย 100% แต่ปลอดภัยพอสำหรับทีมคุณ มีจริง
- Key takeaways
- ก้าวต่อไป

บทที่ 12 — แผน 30 วันสำหรับเริ่ม Local AI Pilot

- ทำไมต้องคิดเป็น “pilot” ไม่ใช่ “ลองดูก่อน”
- สี่หลักที่ทำให้ pilot ตอบคำถามได้จริง
- แผน 30 วัน แบ่งเป็นสี่สัปดาห์
  - สัปดาห์ที่ 1 — วางแผน
  - สัปดาห์ที่ 2 — ลงมือ
  - สัปดาห์ที่ 3 — วัดเทียบ Cloud
  - สัปดาห์ที่ 4 — ตัดสินใจ
- เกณฑ์ตัดสินใจ Go / Pivot / Stop
- ถ้า pilot ไม่เป็นไปตามคาด ทำต่อยังไง
- สามตัวอย่าง สามทางที่ valid
- 30-Day Local AI Pilot Plan (worksheet กรอกได้)
  - ทดสอบกับงานจริงของคุณ
- Mini Project 8 — วางแผน pilot ของตัวเอง
- Key takeaways
- ก้าวต่อไป

ภาคผนวก ก: สร้าง mini-ChatGPT web UI ใช้เอง (Gradio)

- “แต่ผมมี Open WebUI อยู่แล้วนี่ — ทำไมต้องสร้างเอง”
- Gradio คืออะไร เมื่ออธิบายเป็นภาษาคน
- หัวใจของ gr.ChatInterface: ฟังก์ชันตอบหนึ่งตัว
- โค้ดเต็ม ~30 บรรทัด
- ทำไมต้อง yield “ทั้งก้อน” ไม่ใช่แค่ตัวอักษรใหม่
- ทุกกล่องอยู่ในเครื่อง: เส้นทางของข้อความ
- ข้อควรระวัง: share=True คือการเปิดประตูบ้าน
- Gradio, Open WebUI หรือ Streamlit — เลือกตัวไหน
  - Streamlit สำหรับคนที่คุ้นอยู่แล้ว
- ข้อผิดพลาดที่พบบ่อย และวิธีแก้
  - Checklist: รันไม่ได้ ทำไง
- ต่อยอดง่าย ๆ: เปลี่ยน persona และเลือก model
- ทดสอบกับงานจริงของคุณ
- Key takeaways
- ก้าวต่อไป

ภาคผนวก ข: สร้าง Document Q&A web app กับเอกสารของคุณ

- คลังสินค้ากับหน้าร้าน: ทำไมต้องแยกเป็นสองไฟล์
- ไฟล์ที่หนึ่ง: ingest.py — จัดเอกสารเข้าคลัง
- ไฟล์ที่สอง: app.py — หน้าร้านที่ค้นเอกสารก่อนตอบ
- ลำดับการรันทั้งหมด
- สามจุดด้านความปลอดภัยที่ต้องระวังเป็นพิเศษ
- ทดสอบกับงานจริงของคุณ
- ข้อผิดพลาดที่พบบ่อย และวิธีแก้
  - Checklist: รันไม่ได้ / ผลแปลก ทำไง
  - Checklist: ก่อนเปิดให้ทีมใช้งานจริง
- ตอนนี้เทียบกับตอนอยู่ใน terminal
- Key takeaways
- ก้าวต่อไป

ภาคผนวก ค: สร้าง Code Helper CLI ของตัวเอง

ทำไมต้องเขียนเอง ในเมื่อมี Continue.dev แล้ว

เครื่องมือนี้ทำงานยังไง มองจากภายนอกก่อน

โครงสร้างคำสั่งแบบ subcommand ด้วย argparse

ไล่อ่านโค้ดทั้งตัว ทีละส่วน

ลองใช้งานจริง สี่คำสั่ง

เรื่องที่ต้องพูดให้เข้าใจของข้อจำกัดของโมเดลในเครื่อง

ทำไมเครื่องมือนี้ถึง “อ่านอย่างเดียว” โดยตั้งใจ

เช็กลิสต์: รันให้ผ่าน และแก้ตอนติดขัด

Key takeaways

บทนำ

ทุกวันนี้เราเอางานสำคัญไปให้ AI ช่วยมากขึ้นเรื่อย ๆ — สรุปสัญญา ร่างอีเมล อ่านโค้ด วิเคราะห์เอกสารภายใน แล้ววันหนึ่งก็มีคำถามง่าย ๆ แต่ตอบไม่เต็มปากโผล่ขึ้นมา: "ข้อมูลที่เราพิมพ์เข้าไป มันไปจบที่ไหน?" หนังสือเล่มนี้เกิดจากคำถามนั้นครับ

Local AI คือการรันโมเดล AI ไว้ในเครื่องของคุณเอง — โน้ตบุ๊ก พีซี หรือเซิร์ฟเวอร์ในออฟฟิศ — แทนที่จะส่งข้อมูลออกไปประมวลผลบนเซิร์ฟเวอร์ของผู้ให้บริการที่ไหนสักแห่ง หน้าตาการใช้งานเกือบเหมือนเดิม แต่ "แหล่งของคำตอบ" ย้ายมาอยู่ในมือคุณ และนั่นเปิดทางให้สามเรื่องที่คนใช้ Cloud AI จริงจังมักเริ่มต้องการ: คุมข้อมูลให้ไม่ออกนอกเครื่อง ลดค่าใช้จ่ายต่อการเรียกใช้ และปรับแต่งได้อิสระขึ้น

แต่ผมอยากพูดตรง ๆ ตั้งแต่ต้น — Local AI ไม่ใช่ยาวิเศษ และไม่ได้ "ปลอดภัยกว่า" โดยอัตโนมัติ มันคืออีกเครื่องมือหนึ่งที่มีจุดแข็งจุดอ่อนต่างจาก Cloud หนังสือเล่มนี้จะไม่ขายฝัน แต่จะช่วยให้คุณเลือกใช้แต่ละแบบให้ถูกกับงาน และลงมือสร้างของจริงได้ด้วยตัวเอง

เราจะเดินไปด้วยกันเป็นห้าช่วง เริ่มจาก **ทำไม** (บทที่ 1–2 ว่าด้วยเหตุผล และความต่างของ Cloud / Local / Hybrid) ต่อด้วย **เข้าใจ** (บทที่ 3–4 งานไหนควรอยู่ที่ไหน และเรื่อง hardware, ขนาดโมเดล, ต้นทุนที่ต้องรู้ก่อนเริ่ม) จากนั้น **เตรียมเครื่องมือ** (บทที่ 5) แล้วเข้าสู่ช่วง **ลงมือ** ของจริงทีละขั้น — เปิดแชตในเครื่อง (บทที่ 6), เรียกผ่าน Python และ API (บทที่ 7), ทำงานกับเอกสารภายในด้วย RAG (บทที่ 8), ใช้ช่วยงานโค้ด (บทที่ 9) และสร้าง agent เบื้องต้น (บทที่ 10) — ก่อนปิดท้ายที่ช่วง **ขยายผล** สู่การใช้ในองค์กรอย่างปลอดภัย (บทที่ 11) และแผน 30 วันสำหรับเริ่ม pilot (บทที่ 12) ส่วนภาคผนวก ก–ค มีโปรเจกต์เล็กพร้อมโค้ดให้ลองสร้าง mini-ChatGPT, Document Q&A และ Code Helper ไว้ใช้เอง

หนังสือออกแบบให้อ่านได้สองสาย ถ้าคุณไม่เขียนโค้ด ข้ามกล่องโค้ดไปได้เลยโดยยังเห็นภาพรวมครบ ถ้าคุณเป็นนักพัฒนาหรือสาย IT จะมีโค้ดและรายละเอียดให้ลงมือต่อได้ทันที ทุกบทเปิดด้วยสถานการณ์จริง อธิบายเป็นภาษาคนก่อนค่อยลงศัพท์เทคนิค และปิดด้วยข้อสรุปกับก้าวต่อไปเสมอ

เรื่องหนึ่งที่อยากฝากไว้ก่อนเริ่ม — วงการนี้เปลี่ยนเร็วมาก ชื่อเครื่องมือ ราคา และสเปกของโมเดลในเล่มนี้คือภาพ ณ เวลาที่เขียน สิ่งที่ผมตั้งใจมอบให้จริง ๆ คือ "วิธีคิด" ที่ยังใช้ได้แม้รายละเอียดจะเปลี่ยนไป เพราะเหตุผลที่คุณอยากคุมข้อมูลและต้นทุนของตัวเอง จะไม่เปลี่ยนไปไหน

ภาคลงมือทำที่อยากให้ลองจริง — ท้ายเล่มมีภาคผนวกสามชุดที่ออกแบบมาให้ "ลงมือทำ" โดยเฉพาะ พร้อมโค้ดเต็มที่ก็อปไปรันได้ทันที — **ภาคผนวก ก** พาสร้าง mini-ChatGPT web UI ด้วย Gradio, **ภาคผนวก ข** สร้าง Document Q&A app ที่ค้นเอกสารของคุณเองด้วย RAG และ **ภาคผนวก ค** สร้าง Code Helper CLI ไว้ช่วยอ่านและอธิบายโค้ด ถ้าคุณเป็นสายเรียนรู้ผ่านการลงมือทำจริง จะกระโดดมาลองภาคผนวกก่อนแล้วค่อยย้อนไปอ่านบทที่เกี่ยวข้องก็ได้

พร้อมแล้วก็เริ่มกันที่คำถามแรกเลยครับ — ทำไมคนที่ใช้ Cloud AI อยู่ดี ๆ ถึงเริ่มหันมามอง Local AI

บทที่ 1 — ทำไมคนใช้ Cloud AI ถึงเริ่มสนใจ Local AI

บ่ายสามวันธรรมดา หัวหน้าเดินมาที่โต๊ะคุณ ถือมือถือมาด้วย แล้วถามว่า

"เมื่อกี้ที่เอาสัญญาลูกค้าไปให้ AI ช่วยสรุปน่ะ — มันเก็บข้อมูลเราไหม?"

คุณหยุดคิดครู่หนึ่ง แล้วตอบไปว่า "...น่าจะไม่นะครับ" แต่พอพูดจบ คุณก็รู้ทันทีว่าตัวเองตอบไม่ได้เต็มปาก เพราะความจริงคือ คุณไม่แน่ใจ คุณใช้ AI ตัวนี้สรุปงานมาเป็นเดือน มันช่วยประหยัดเวลาคุณได้มหาศาล แต่คำถามง่าย ๆ แค่ว่า "ข้อมูลที่เราป้อนเข้าไป มันไปจบที่ไหน" — คุณกลับตอบไม่ได้

พอหัวหน้าเดินจากไป คุณเปิดหน้า Privacy Policy ของ provider ที่ใช้อยู่ขึ้นมาอ่าน เลื่อนไปสักพัก แล้วก็เจอว่า — มันยาวมาก เต็มไปด้วยศัพท์ และที่สำคัญคือ มันขึ้นอยู่กับว่าคุณใช้ "แพลนไหน" ซึ่งคุณก็จำไม่ได้แล้วว่าทีมสมัครแพลนอะไรไว้ บางบรรทัดบอกว่าข้อมูลอาจถูกนำไปปรับปรุงบริการ บางบรรทัดบอกว่ามีทางปิด แต่ปุ่มนั้นอยู่ตรงไหนก็ไม่รู้

ถ้าฉากนี้คุ้น ๆ — คุณไม่ได้เป็นคนขี้กลัว AI และนี่ก็ไม่ใช่ความหวาดระแวงเกินเหตุ คุณแค่เริ่มถามคำถามที่คนใช้ AI จริงจังทุกคนจะถามในที่สุด พอคุณเอาเครื่องมือมาทำงานกับข้อมูลที่สำคัญขึ้นเรื่อย ๆ คำถามที่ตามมาตามธรรมชาติก็คือ "แล้วข้อมูลพวกนี้ ควรอยู่กับใคร"

ผมเขียนหนังสือเล่มนี้ขึ้นมาเพราะคำถามนั้นแหละครับ ก่อนที่เราจะไปถึงเรื่องว่า "สร้าง Local AI ยังไง" ผมอยากให้เราตอบคำถามของหัวหน้าให้เต็มปากก่อน เพราะคำตอบที่ถูกต้อง ไม่ใช่ "ใช่" และไม่ใช่ "ไม่ใช่"

คำตอบคือ **"ขึ้นอยู่กับว่าคุณใช้ tier ไหน และปิดการตั้งค่าไว้หรือยัง"**

คำตอบที่หัวหน้าอยากได้จริง ๆ — มันขึ้นกับ tier

คำว่า tier คือ "ระดับแพลน" ที่คุณสมัครใช้ ผู้ให้บริการ Cloud AI รายใหญ่แทบทุกเจ้าแบ่งบริการเป็นหลายชั้น และหัวใจที่คนส่วนใหญ่ไม่รู้คือ — **แต่ละ tier มีนโยบายข้อมูล "ค่าเริ่มต้น" ต่างกันมาก**

พูดกว้าง ๆ ให้เห็นภาพ (และขอให้เข้าใจตรงกันว่าผมพูดเชิงหลักการ ไม่ใช่ฟันธงแทนทุกเจ้า):

- **แพลนแบบ consumer** (แพลนฟรีหรือรายบุคคลที่คุณสมัครเองผ่านเว็บ) — บางเจ้า "ค่าเริ่มต้น" คือนำบทสนทนาของคุณไปช่วยพัฒนาโมเดล เว้นแต่คุณจะเข้าไปปิดเองในหน้า settings ปัญหาคือ ผู้ใช้ทั่วไปมักไม่รู้ว่าต้องปิด หรือไม่รู้ว่าปุ่มนั้นอยู่ตรงไหน
- **แพลนแบบ business / enterprise / API** — โดยทั่วไปจะ "ไม่นำ" ข้อมูลของคุณไปเทรนโมเดลตั้งแต่ค่าเริ่มต้น เพราะมีสัญญาเชิงพาณิชย์กำกับอยู่

แล้วทำไมค่าเริ่มต้นถึงต่างกันขนาดนี้ — สำหรับคนสายธุรกิจ มันมีเหตุผลที่เข้าใจได้ไม่ยากเลยครับ แพลน enterprise และ API มักมาพร้อมข้อตกลงเชิงพาณิชย์และสัญญาประมวลผลข้อมูล (data processing agreement หรือ DPA) ที่ระบุชัดว่า provider จะทำอะไรกับข้อมูลของลูกค้าได้บ้าง เพราะลูกค้าองค์กรจ่ายเงิน

และต้องการความมั่นใจทางกฎหมายก่อนเอาข้อมูลธุรกิจเข้าระบบ ส่วนแพลน consumer ที่ใช้ฟรีหรือจ่ายน้อย ในมุมหนึ่งคือการแลกเปลี่ยน — คุณได้ใช้บริการในราคาถูกหรือฟรี แลกกับการที่ข้อมูลบางส่วนอาจถูกนำไปช่วยพัฒนาบริการต่อ เว้นแต่คุณจะเข้าไปปิดเอง

นี่ไม่ใช่เรื่องของ provider ที่ดีหรือร้าย แต่เป็นเรื่องของ "โมเดลธุรกิจ" ที่ต่างกันคนละชั้น เข้าใจตรงนี้แล้วคุณจะเลือกแพลนให้ตรงกับความอ่อนไหวของข้อมูลได้ดีขึ้น (ย้ำว่าผมพูดเชิงหลักการ — รายละเอียดจริงของแต่ละเจ้าให้ไปตรวจหน้า terms ของเขาเอง)

ความต่างนี้สำคัญมาก เพราะมันแปลว่า "Cloud AI" ไม่ใช่ก้อนเดียวกันทั้งหมด การพูดว่า "Cloud AI เก็บข้อมูลคุณไปเทรน" จึงผิด พอ ๆ กับการพูดว่า "Cloud AI ปลอดภัย 100%" — ความจริงอยู่ตรงกลาง และอยู่ที่รายละเอียดของ tier

ขอยกตัวอย่างที่จับต้องได้หนึ่งอัน เพราะมันสะท้อนประเด็นได้ดี ตามนโยบายของ Google Gemini แบบ consumer ที่ตรวจพบ ณ ช่วงเวลาที่เขียนหนังสือเล่มนี้ — บทสนทนาที่ถูกทีมงานนำไป review ไว้แล้ว อาจถูกเก็บไว้ได้นานถึง 3 ปี **แม้คุณจะกดลบบทสนทนานั้นไปแล้วก็ตาม** (ส่วนถ้าปิดการเก็บ activity ระยะเก็บจะเหลือ 72 ชั่วโมง)

จุดที่ผมอยากให้จำไม่ใช่ตัวเลข "3 ปี" แต่เป็นบทเรียนเบื้องหลังมัน — บนแพลตฟอร์ม consumer คำว่า "ลบแล้ว" ไม่ได้แปลว่า "ข้อมูลหายทันที" เสมอไป

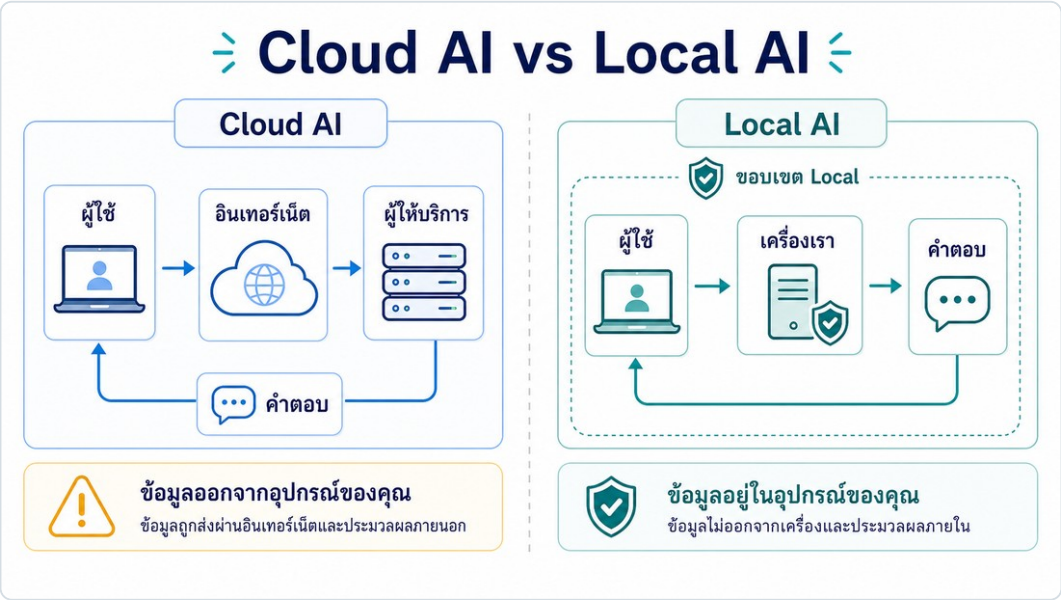
> **ข้อควรระวังสำคัญ:** นโยบายข้อมูลของผู้ให้บริการทุกเจ้าเปลี่ยนบ่อยมาก และมักไม่มีการแจ้งเตือนผู้ใช้ชัด ๆ ตัวเลขและรายละเอียดทุกอย่างในหัวข้อนี้ "ตรวจ ณ เวลาเขียน" — ถ้าคุณจะตัดสินใจอะไรจริงจัง ให้เข้าไปอ่านหน้า privacy ของ provider ที่คุณใช้อยู่โดยตรงอีกครั้งเสมอ

ทีนี้ คุณตอบหัวหน้าได้เต็มปากขึ้นแล้ว — "ขึ้นกับว่าทีมเราใช้แพลนไหน เดี๋ยวผมเช็กให้ และจะปิดการตั้งค่าที่ควรปิด" นั่นคือก้าวแรก แต่สำหรับบางคน บางองค์กร คำถามถัดมาคือ — แล้วถ้าเราไม่อยากต้องลุ้นกับเรื่องพวกนี้เลยล่ะ? ข้อมูลบางชุดเราอยากให้มัน "ไม่ออกไปไหนเลย" ได้ไหม?

นั่นแหละครับ คือจุดที่ Local AI เริ่มเข้ามาในภาพ

> **Key idea:** คำถาม "AI เก็บข้อมูลเราไหม" ไม่มีคำตอบเดียว — มันขึ้นกับ tier และการตั้งค่า สิ่งที่ Local AI เสนอ ไม่ใช่ "ปลอดภัยกว่าเสมอ" แต่คือ "ไม่ต้องลุ้นว่าข้อมูลวิ่งไปไหน เพราะมันไม่ได้วิ่งออกไป"

Local AI คืออะไร แบบสั้นที่สุด



ก่อนไปต่อ ขออธิบายให้ตรงกันก่อน เผื่อบางคนยังไม่เคยได้ยินคำนี้

Local AI หมายถึงการรันโมเดล AI ไว้ในเครื่องของคุณเอง — โน้ตบุ๊ก พีซี หรือเซิร์ฟเวอร์ในออฟฟิศ — แทนที่จะเรียกใช้บริการที่รันอยู่บนเซิร์ฟเวอร์ของผู้ให้บริการที่ไหนสักแห่งบนอินเทอร์เน็ต

ลองนึกภาพแบบนี้ ตอนนี้เวลาคุณพิมพ์คุยกับ ChatGPT หรือ Claude ผ่านเว็บ ข้อความของคุณวิ่งออกจากเครื่อง ไปประมวลผลที่เซิร์ฟเวอร์ของ provider แล้วคำตอบวิ่งกลับมา ส่วน Local AI — ข้อความนั้นไม่ได้วิ่งไปไหนเลย มันถูกประมวลผลในเครื่องคุณ แล้วตอบกลับมาในเครื่องคุณ หน้าตาการใช้งานคล้ายเดิมมาก แค่ "แหล่งของคำตอบ" ย้ายมาอยู่ในมือคุณ

ถ้าคุณเคยอ่าน**เล่มพื้นฐาน** (คู่มือเริ่มต้นเป็น AI Engineer อย่างเป็นระบบ) มาก่อน คุณจะคุ้นกับแนวคิด "เรียกใช้ LLM ผ่าน API ของ provider" ทันที — เล่มนี้ก็คือการต่อยอดจากตรงนั้น เปลี่ยนจาก "เรียก AI ที่อยู่บน cloud" มาเป็น "เรียก AI ที่รันอยู่ในเครื่องคุณเอง" ส่วนถ้าคุณยังไม่เคยอ่าน ก็ไม่เป็นไรเลย เล่มนี้อ่านจบได้ด้วยตัวมันเอง ผมจะแทรกสิ่งที่ต้องรู้ให้ตามทาง

เหตุผลจริง ๆ ที่คนเริ่มมอง Local AI

ทีนี้ลองมาดูกันว่า คนที่ใช้ Cloud AI คล่องอยู่แล้ว เริ่มหันมามอง Local AI เพราะอะไรบ้าง — ไม่ใช่เพราะกระแส แต่เพราะเจอปัญหาจริงในงาน ผมสรุปได้เป็น 5 ข้อ

1. ความเป็นส่วนตัวและการคุมข้อมูล (Privacy & Data Control) — นี่คือแรงจูงใจอันดับหนึ่งในทางปฏิบัติ ประเด็นไม่ใช่ว่า provider เป็นคนไม่ดี แต่เป็นเรื่องเชิงโครงสร้าง — ตราบใดที่ข้อมูลของคุณวิ่งผ่านเซิร์ฟเวอร์ที่ "ไม่ใช่ของคุณ" คุณก็ต้องเชื่อใจระบบที่คุณคุมไม่ได้ Local AI ตัดข้อกังวลนี้ออกได้ในเชิงโครงสร้าง เพราะข้อมูลตอนประมวลผลไม่วิ่งออกไปหาเซิร์ฟเวอร์ภายนอกเลย ลองนึกถึงฝ่าย HR ที่ต้องสรุปใบสมัครงานหลายสิบไฟล์ต่อสัปดาห์ ในไฟล์เต็มไปด้วยข้อมูลส่วนบุคคลที่คุ้มครองโดย PDPA — งานเอกสารซ้ำ ๆ บวกกับข้อมูลส่วนบุคคล คือสูตรที่ Local AI ตอบโจทย์ได้ดี

2. ต้นทุนที่คาดเดาได้ (Cost Predictability) — ปัญหาจริงของหลายทีมไม่ใช่ "Cloud แพงเกินไป" แต่เป็น "Cloud แพงแบบไม่รู้ล่วงหน้า" เพราะคิดเงินตามปริมาณการใช้ ยิ่งใช้ยิ่งจ่าย และพอคุณเริ่มต่อ AI เข้ากับ workflow อัตโนมัติหรือ agent งานหนึ่งงานอาจสร้างการเรียกใช้หลายรอบโดยที่คุณไม่ทันสังเกต ทำให้ค่าใช้จ่ายพุ่งขึ้นได้หลายเท่าในเวลาไม่กี่เดือน — ผลสำรวจในวงการ (ซึ่งเป็นของผู้ขายบริการ จึงต้องรับฟังด้วยความระวัง) ก็สะท้อนทิศทางนี้เช่นกัน Local AI ไม่คิดเงินต่อการเรียกใช้ จึงให้ต้นทุนที่นิ่งกว่าสำหรับงานปริมาณมากและสม่ำเสมอ

3. การไม่พึ่งพาเจ้าเดียว (Vendor Independence) — เมื่อคุณวาง workflow สำคัญทั้งหมดไว้บน provider เจ้าเดียว คุณก็ผูกชะตากับเขา ถ้าวันหนึ่งเขาขึ้นราคา เปลี่ยนเงื่อนไข หรือปลดรุ่นโมเดลที่คุณใช้อยู่ กระบวนการของคุณอาจสะดุดทันที ความกังวลเรื่องนี้ในหมู่องค์กรนั้นมีจริงและกว้างขวาง (ผลสำรวจหลายชิ้นในวงการชี้ไปทางเดียวกัน แม้ส่วนใหญ่จะมาจากผู้ขายที่มีส่วนได้เสีย จึงควรมองเป็นแนวโน้ม ไม่ใช่ตัวเลขฟันธง) Local AI ช่วยลดการผูกขาดนี้ได้ส่วนหนึ่ง — ไม่ใช่ทางเดียว แต่เป็นทางหนึ่ง

4. ทำงานได้แม้ไม่มีเน็ต (Offline / Independence) — บางสภาพแวดล้อมตัดขาดจากอินเทอร์เน็ตโดยตั้งใจ เช่น ระบบที่ต้องการความมั่นคงสูง โรงงานบางแห่ง หรือพื้นที่ที่เน็ตไม่เสถียร Local AI ทำงานได้โดยไม่ต้องพึ่งการเชื่อมต่อออกนอก และโดยธรรมชาติของมัน — เพราะประมวลผลในเครื่อง คำขอจึงไม่ต้องวิ่งออกไปกลับผ่านเครือข่าย ไม่มี network round-trip มาคั่นกลางเหมือนตอนเรียกบริการบน cloud

5. การเรียนรู้และทดลอง — กลุ่มนี้มักเป็น developer และคนที่อยากเข้าใจ AI ลึกขึ้น การรันในเครื่องเองทำให้คุณทดลองโมเดล ปรับ prompt เปรียบเทียบรุ่นต่าง ๆ ได้ไม่จำกัดครั้ง โดยไม่มีค่าใช้จ่ายต่อการเรียกใช้ — เหมาะมากสำหรับคนที่อยากลองซ้ำ ๆ จนเข้าใจ

> **Key idea:** เหตุผลที่คนหันมามอง Local AI ส่วนใหญ่ไม่ใช่ "อยากเลิกใช้ Cloud" แต่เป็น "อยากได้ทางเลือกสำหรับบางงาน" ที่ Cloud ตอบให้ไม่ได้ — โดยเฉพาะงานที่ข้อมูลอ่อนไหว ต้นทุนต้องนิ่ง หรือต้องทำงานแบบ offline

ก่อนไปต่อ ขออธิบาย 3 ความเข้าใจผิดที่ต้องเคลียร์

ก่อนที่คุณจะเดินหน้าด้วยความคาดหวังผิด ๆ ผมขอวางความคาดหวังให้ตรงก่อน เพราะถ้าเริ่มต้นด้วยความเข้าใจที่บิด คุณจะผิดหวังกลางทางแล้วเลิกไปเปล่า ๆ มี 3 เรื่องที่ผมเจอคนเข้าใจผิดบ่อยที่สุด

ความเข้าใจผิดที่ 1: "Local AI ฟรี"

ผมต้องบอกตรง ๆ ก่อนว่า — Local AI ไม่ได้ฟรี มันแค่ "เปลี่ยนรูปแบบต้นทุน" ไม่ใช่ "ตัดต้นทุนทิ้ง" จริงอยู่ว่าไม่มีค่าจ่ายต่อการเรียกใช้ และเครื่องมืออย่าง Ollama ก็โหลดมาใช้ได้ฟรี แต่ต้นทุนมันย้ายไปอยู่ที่อื่น — ค่า hardware (ถ้าต้องซื้อใหม่), ค่าไฟ, และที่คนมองข้ามที่สุดคือ "เวลาและแรงในการดูแล" ทั้งติดตั้ง อัปเดต และแก้ปัญหา

ตรงนี้ต้องแยกสเกลให้ชัด สำหรับการ deploy ระดับองค์กรจริงจัง ต้นทุนรวมเมื่อนับค่าแรงวิศวกรและ overhead ทั้งหมดมักสูงกว่าราคา hardware ที่เห็นหลายเท่า **แต่** — และนี่สำคัญมาก — สำหรับคุณที่เพิ่งเริ่ม ใช้โน้ตบุ๊กหรือพีซีที่มีอยู่แล้ว ต้นทุนเพิ่มเติมแทบจะมีแค่ "ค่าไฟกับเวลา" เท่านั้น ตัวเลขค่าใช้จ่ายมหาศาลที่คุณอาจเคยเห็นในบทความ มันคือสเกลองค์กร ไม่ใช่สเกลของคุณที่กำลังลองบนเครื่องตัวเอง อย่าเอามาขู่ตัวเอง

ความเข้าใจผิดที่ 2: "Local AI เก่งกว่า / ดีกว่า Cloud AI"

ความจริงที่ไม่ค่อยมีใครพูดคือ — ในหลายงาน frontier model บน cloud ยังเก่งกว่าโมเดล open-weight ที่รันบนเครื่องทั่วไปอยู่พอสมควร โดยเฉพาะงานคิดหลายชั้นที่ซับซ้อน งานเขียนโค้ดยาก ๆ หลายไฟล์ งานที่เกี่ยวกับภาพ-เสียง-วิดีโอ และ agent ที่ต้องแก้ปัญหาเองหลายขั้น ช่องว่างนี้ ณ ตอนที่เขียนอยู่ราว ๆ ไม่กี่เดือนบน benchmark ทั่วไป และกำลังแคบลงเร็ว แต่ตอนนี้มันยังมีอยู่จริง

ขณะเดียวกัน สำหรับงานเฉพาะหลายอย่าง — สรุปเอกสาร ถามตอบจากไฟล์ภายใน อธิบาย error พื้นฐานในโค้ด จัดหมวดหมู่ซ้ำ ๆ — โมเดล local ก็ "เพียงพอ" แล้ว คำถามที่ถูกจึงไม่ใช่ "อันไหนเก่งกว่า" แต่เป็น **"งานนี้ต้องการความสามารถระดับ frontier จริงไหม"**

ความเข้าใจผิดที่ 3: "รันได้ = ใช้งานได้จริง"

ขอเคลียร์ตรงนี้เลย — มีระยะห่างระหว่าง "โมเดลตอบได้" กับ "ระบบที่ใช้งานได้จริงในงานของคุณ" tutorial หลายอันทำให้ดูเหมือนติดตั้งเสร็จใน 5 นาทีแล้วใช้ได้เลย ความจริงคือ บนเครื่องที่ไม่มีการ์ดจอช่วย โมเดลขนาดกลางอาจใช้เวลาตอบ 10–30 วินาทีต่อหนึ่งคำตอบ ซึ่งบางงานก็พอรับได้ แต่ถ้าคุณต้องคุยโต้ตอบเร็ว ๆ มันช้าเกินจนน่าหงุดหงิด

"รันได้" จึงเป็นแค่จุดเริ่ม ส่วน "ใช้งานได้จริง" ต้องอาศัยการเลือกโมเดลให้เหมาะกับงาน มี hardware ที่พอ และออกแบบ workflow ให้ถูก — ซึ่งก็คือสิ่งที่ทั้งเล่มนี้จะค่อย ๆ พาคุณไปทำ

> **Key idea:** Local AI ไม่ได้ฟรี ไม่ได้เก่งกว่าเสมอ และ "รันได้" ไม่เท่ากับ "ใช้งานได้จริง" — แต่มันตอบโจทย์งานบางอย่างที่ Cloud ให้ไม่ได้ ถ้าคุณเริ่มด้วยความเข้าใจที่ตรง คุณจะไม่ผิดหวังกลางทาง

Local ไม่ใช่ตัวแทน Cloud — มันคืออีกเครื่องมือ

มาถึงตรงนี้ ผมอยากวาง framing ที่จะอยู่กับเราไปทั้งเล่ม

เวลาคนได้ยินคำว่า "รัน AI ใช้เอง" หลายคนคิดว่าต้องเลือกข้าง — ย้ายมา Local ทั้งหมด หรือใช้ Cloud ต่อไป จริง ๆ แล้วไม่ใช่เลย

ลองนึกภาพการทำกับข้าว — **Cloud AI ก็เหมือนสั่ง delivery** สะดวก รวดเร็ว เชฟฝีมือดี คุณไม่ต้องมีครัว ไม่ต้องล้างจาน แต่ครัวไม่ใช่ของคุณ คุณไม่รู้ว่าเขาใส่อะไรลงไปบ้าง และของกินผ่านมือคนกลางมาก่อนถึงคุณ ส่วน **Local AI ก็เหมือนทำกับข้าวเองที่บ้าน** คุณคุมวัตถุดิบได้ รู้ว่าใส่เกลือใส่น้ำมันแค่ไหน ของมันอยู่กับคุณตลอด — แต่คุณต้องมีครัว มีเวลา มีฝีมือ และต้องดูแลเอง

(ถ้าคุณเคยอ่านเล่มพื้นฐาน คุณอาจจำ analogy "เชฟที่ไม่ปลูกผักเอง" ได้ — อันนี้คือพี่น้องของมัน)

จุดสำคัญของการเปรียบเทียบนี้คือ — **การมีครัวที่บ้าน ไม่ได้แปลว่าคุณจะเลิกสั่ง delivery ตลอดไป** มันแปลว่าคุณมีทางเลือกสำหรับบางมื้อที่อยากคุมเองมากกว่า บางมื้อทำเองคุ้มและอร่อยพอ บางมื้อก็ยอมรับตรง ๆ ว่าฝีมือเชฟร้านนั้นเหนือกว่า เราก็สั่ง

Local AI ก็อยู่ในหมวดเดียวกัน มันไม่ได้มาแทนที่ Cloud มันมาเป็น **อีกเครื่องมือ** ในกล่องเครื่องมือของคุณ และคำถามที่ดีที่สุดไม่ใช่ "Local หรือ Cloud อันไหนดีกว่า" แต่เป็น **"งานนี้ ข้อมูลนี้ ควรอยู่ที่ไหน"**

มีตัวอย่างจริงที่ผมชอบ ในบทความของเว็บ XDA Developers นักเขียนสายเทคโนโลยีคนหนึ่งเล่าว่า เขาเอาการ์ดจอเก่าที่มีอยู่แล้วในบ้านมารันโมเดล local ขนาดเล็กถึงกลาง ใช้กับงานจัดการเอกสารส่วนตัว ระบบสมาร์ทโฮม และวิเคราะห์โค้ดในโปรเจกต์ที่บ้าน — โดยแทบไม่มีต้นทุนใหม่ แต่เขาก็พูดตรง ๆ ว่า สำหรับงานคิดซับซ้อน บริการ cloud อย่าง ChatGPT, Gemini หรือ Claude ยังนำหน้าโมเดล local ของเขาอยู่หลายขุม นี่แหละครับ คือ mindset ที่ถูก — เริ่มจากของที่มี ใช้ local กับงานที่มันเหมาะ และยอมรับตรง ๆ ว่ามันไม่ใช่คำตอบของทุกงาน

> **Key idea:** "ครัวที่ดีไม่ได้แปลว่าคุณจะเลิกสั่ง delivery" — Local AI ไม่ได้ตอบคำถามว่าจะใช้ Cloud ต่อหรือเปล่า มันตอบคำถามว่า "งานนี้ ข้อมูลนี้ ควรอยู่ที่ไหน"

Cloud คุมอะไรได้/ไม่ได้ vs Local คุมอะไรได้/ต้องดูแลอะไร

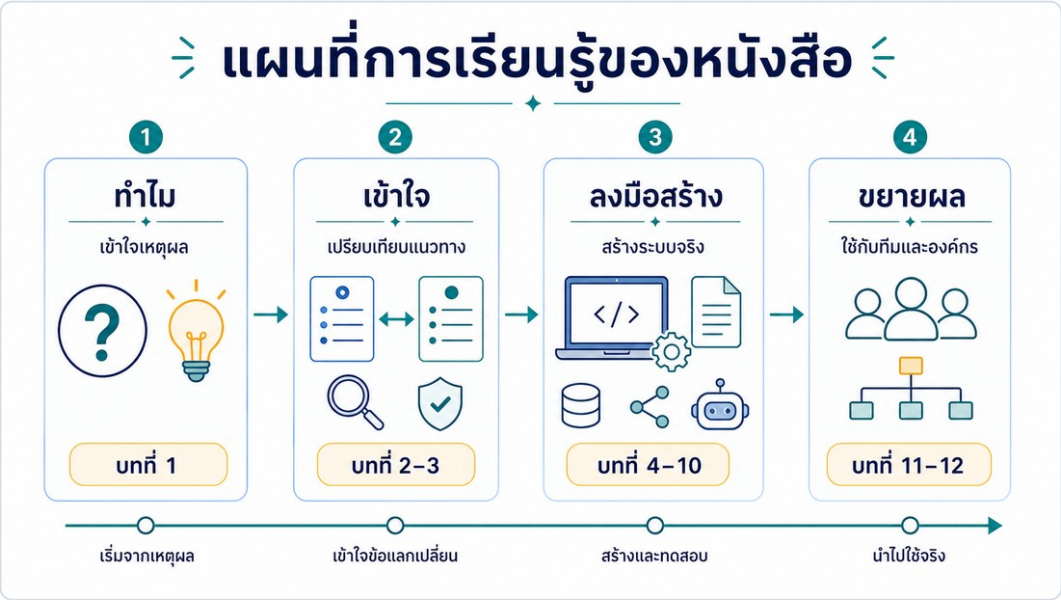
เพื่อให้เห็นภาพรวมว่าจริง ๆ แล้ว "ความรับผิดชอบ" มันย้ายไปอยู่ตรงไหนบ้างเมื่อคุณเลือก Local ผมสรุปเป็นตารางสั้น ๆ ไว้ สังเกตว่าผมแยก Cloud ออกเป็นสองคอลัมน์ เพราะอย่างที่บอกไปแล้ว — consumer tier กับ enterprise tier ต่างกันมากจนรวมกันไม่ได้

มิติ	Cloud (consumer tier)	Cloud (API / enterprise)	Local AI
ข้อมูลตอนประมวลผลวิ่งไปไหน	เซิร์ฟเวอร์ provider (อาจอยู่ต่างประเทศ)	เซิร์ฟเวอร์ provider (เลือก region ได้ในบางเจ้า)	อยู่ในเครื่อง/เซิร์ฟเวอร์ของคุณ
นำข้อมูลไปเทรนโมเดลไหม	บางแพลนมี โดยค่าเริ่มต้น (ปิดเองได้)	โดยทั่วไปไม่	ไม่ (ตัวเครื่องมือไม่เทรน)
ใครดูแล infrastructure	provider	provider	คุณดูแลเอง
รูปแบบต้นทุน	จ่ายต่อการใช้ / รายเดือน	จ่ายต่อการใช้ / รายเดือน	hardware + ค่าไฟ + เวลาดูแล
ความสามารถของโมเดล	ระดับ frontier	ระดับ frontier	จำกัดกว่า frontier แต่พอสำหรับหลายงาน
ใช้แบบ offline ได้ไหม	ไม่ได้	ไม่ได้	ได้
ต้องดูแล security เองไหม	ส่วนใหญ่ provider จัดการ	ส่วนใหญ่ provider จัดการ	คุณต้องจัดการเอง

อ่านตารางนี้แล้วจะเห็นว่า การย้ายมา Local ไม่ใช่ "ได้ทุกอย่างฟรี ๆ" — สิ่งที่ provider เคยแบกให้ (การดูแลเครื่อง การคุม security) ย้ายมาอยู่บนบ่าคุณ คุณได้การควบคุมข้อมูลกลับมา แลกกับความรับผิดชอบที่เพิ่มขึ้น นั่นคือดีลของ Local AI และมันเป็นดีลที่คุ้มหรือไม่ — ขึ้นกับงานของคุณ ซึ่งบทที่ 2 และ 3 จะช่วยคุณตัดสินใจอย่างเป็นระบบ

หนังสือเล่มนี้จะพาคุณไปถึงไหน

ก่อนจะลงมือ ผมอยากให้คุณเห็นแผนที่ทั้งเล่ม จะได้รู้ว่าตอนนี้เรายืนอยู่ตรงไหน และกำลังจะไปทางไหน นี่คือเส้นทางการเรียนรู้ทั้ง 12 บท




```
[ทำไม]      บท 1  ทำไมต้อง Local AI  (เหตุผล + ความเข้าใจผิด + วาง framing)  ← คุณอยู่ตรงนี้
               |
[เข้าใจ]    บท 2  Cloud / Local / Hybrid ต่างกันยังไง
               |
       บท 3  งานไหนควรอยู่ Local งานไหนควรอยู่ Cloud  (กรอบตัดสินใจ)
               |
[เตรียม]    บท 4  Hardware, Model และต้นทุนที่ต้องรู้
               |
       บท 5  Tool Stack: Ollama, LM Studio, Open WebUI, llama.cpp
               |
[ลงมือ]     บท 6  สร้าง Local AI Chat ใช้เอง
               |
       บท 7  เรียก Local AI ผ่าน Python และ API
               |
       บท 8  Local Document AI: ใช้ AI กับเอกสารภายใน
               |
       บท 9  Local Coding Assistant: ช่วยอ่านโค้ด แก้ error
               |
       บท 10 Local AI Agent: Tools, API, Connector และ MCP
               |
[ขยายผล]    บท 11 ใช้ในองค์กร: Privacy, Security, Governance
               |
       บท 12 แผน 30 วันเริ่ม Local AI Pilot
```

โครงคือ — เริ่มจาก “ทำไม” (บทนี้) → “เข้าใจความต่างและตัดสินใจ” (บท 2–3) → “เตรียมเครื่องและเครื่องมือ” (บท 4–5) → “ลงมือทำจริง” (บท 6–10) → “ขยายสู่องค์กรและวางแผน” (บท 11–12) เครื่องมือที่เราพูดถึงอาจเปลี่ยนรุ่นเปลี่ยนชื่อในอนาคต แต่หลักคิดในแต่ละขั้นจะใช้ได้นาน

เลือกเส้นทางอ่านของคุณ — Reader Path Map

หนังสือเล่มนี้เขียนสำหรับผู้อ่าน 3 กลุ่มที่ต่างกัน คุณไม่จำเป็นต้องอ่านทุกบทเท่ากันหมด ลองหาแถวที่ตรงกับคุณที่สุด แล้วโฟกัสตามนั้น

เส้นทาง	คุณคือใคร	อ่านลึกบทไหน	ข้ามหรืออ่านผ่าน ๆ ได้
สาย Non-Coder	คนทำงานทั่วไป ใช้ AI ช่วยงานเอกสาร ไม่เขียนโค้ด	บท 1, 2, 3, 5 (เฉพาะส่วน no-code), 6, 11, 12	บท 7, 9, 10 (ส่วนโค้ด) — อ่านสรุปท้ายหัวข้อพอ
สาย Developer	dev / automation builder อยากลงมือทำเอง	ทุกบท โดยเฉพาะ 4, 5, 7, 8, 9, 10	บท 11 อ่านเร็วได้ ถ้าไม่ดูแลฝั่ง governance
สาย Business-IT	เจ้าของธุรกิจ / IT ที่ต้องตัดสินใจก่อนลงทุน	บท 1, 2, 3, 11, 12	บท 5–10 อ่านเพื่อเข้าใจภาพรวม ไม่ต้องลงมือทำเอง

ให้ผมขยายความแต่ละสายสั้น ๆ จะได้เห็นภาพว่าคุณจะได้อะไรกลับไป ถ้าคุณเป็น **สาย Non-Coder** เป้าหมายของคุณคือใช้ Local AI ช่วยงานเอกสารและงานประจำได้จริง ผ่านเครื่องมือแบบ no-code ที่ไม่ต้องพิมพ์คำสั่งสักบรรทัด — บทที่มีโค้ดคุณอ่านแค่สรุปก็พอเข้าใจว่ามันทำอะไรได้ ถ้าคุณเป็น **สาย Developer** คุณจะได้ทั้ง mental model และโค้ดจริงที่เอาไปต่อยอดเป็น workflow หรือ agent ของตัวเองได้ และถ้าคุณเป็น **สาย Business-IT** สิ่งที่คุณต้องการคือกรอบตัดสินใจและประเมินความเสี่ยง มากกว่าการลงมือติดตั้งเอง — บทเชิงเทคนิคคุณอ่านเพื่อรู้ว่าทีมต้องทำอะไร ไม่ต้องทำเองทุกขั้น

ทุกบทที่มีโค้ดจะมีสรุปภาษาคนกำกับไว้เสมอ คุณจึงตามเนื้อหาหลักได้ครบ แม้จะไม่แตะโค้ดเลยสักบรรทัด

> **Key idea:** ไม่มีใครต้องอ่านหนังสือเล่มนี้เหมือนกันหมด เลือกเส้นทางที่ตรงกับงานและทักษะของคุณตั้งแต่ตอนนี้ แล้วคุณจะได้ประโยชน์เร็วขึ้นโดยไม่เสียเวลากับส่วนที่ยังไม่จำเป็น

คุณควรลอง Local AI ไหม — เช็กตัวเองก่อน

ก่อนปิดบท ลองตอบ 7 ข้อนี้ด้วยใจที่ซื่อสัตย์ ตอบแค่ "ใช่" หรือ "ไม่ใช่" แล้วนับจำนวนข้อที่ตอบว่า "ใช่"

เช็กลิสต์: คุณควรลอง Local AI ไหม

สำหรับทุกคน

- [ ] ผมมีงานที่ใช้ AI ช่วยซ้ำ ๆ เป็นประจำ (เช่น สรุป meeting ทุกสัปดาห์ อ่านไฟล์ทุกวัน)
- [ ] ผมมีข้อมูลที่ "ไม่กล้า" เอาขึ้น cloud เพราะ policy หรือความเป็นส่วนตัว
- [ ] ค่า AI tool รายเดือนของผมหรือทีม เริ่มสูงขึ้นและคาดเดายาก
- [ ] ผมมีเครื่อง (โน้ตบุ๊ก/พีซี) ที่แรมตั้งแต่ 16 GB ขึ้นไป ที่พร้อมเอามาลองได้

สำหรับสาย Developer / IT เพิ่มอีก

- [ ] ผมมีโค้ดหรือข้อมูลภายในที่ policy ห้ามออกนอกองค์กร
- [ ] ผมอยากเข้าใจว่า AI ทำงานยังไงจริง ๆ โดยควบคุม environment เองได้

สำหรับสาย Business / องค์กร เพิ่มอีก

- [ ] ผมต้องการ AI ที่ทำงานได้แม้ไม่มีอินเทอร์เน็ต หรือในระบบที่จำกัดการเชื่อมต่อ

อ่านผลแบบนี้

- **ตอบ "ใช่" ตั้งแต่ 3 ข้อขึ้นไป** — Local AI น่าลองสำหรับคุณ บทต่อ ๆ ไปจะพาคุณเริ่มทีละขั้น
- **ตอบ "ใช่" น้อยกว่า 3 ข้อ** — อาจยังไม่ถึงเวลา ลองอ่านถึงบท 3 ก่อนค่อยตัดสินใจอีกที ก็ยังไม่สาย

ไม่ว่าผลจะออกมาแบบไหน นี่ไม่ใช่ข้อสอบที่มีผิดมีถูก มันคือเครื่องมือให้คุณรู้จักงานและข้อมูลของตัวเองชัดขึ้น — ซึ่งเป็นทักษะที่สำคัญกว่าการรู้จัก tool ตัวไหน

Key takeaways

- คำถาม "AI เก็บข้อมูลเราไหม" ไม่มีคำตอบเดียว — มันขึ้นกับ **tier** ที่ใช้และการตั้งค่า บนแพลน consumer บางเจ้า "ลบแล้ว" ไม่ได้แปลว่าหายทันที ส่วนแพลน enterprise/API มักป้องกันเข้มกว่ามาก (ตรวจนโยบายล่าสุดเองเสมอ)
- คนหันมามอง Local AI ด้วยเหตุผลจริง 5 ข้อ: คุมข้อมูล, ต้นทุนคาดเดาได้, ไม่พึ่งเจ้าเดียว, ทำงาน offline, และเรียนรู้-ทดลอง
- มี 3 ความเข้าใจผิดที่ต้องเคลียร์ก่อน: Local **ไม่ฟรี** (เปลี่ยนรูปแบบต้นทุน ไม่ใช่ตัดทิ้ง), **ไม่ได้เก่งกว่า Cloud เสมอ** (frontier ยังนำในงานยาก), และ **"รันได้" ไม่เท่ากับ "ใช้งานได้จริง"**
- Local ไม่ใช่ตัวแทน Cloud แต่เป็น **อีกเครื่องมือ** — คำถามที่ถูกคือ "งานนี้ ข้อมูลนี้ ควรอยู่ที่ไหน" ไม่ใช่ "อันไหนดีกว่า"
- เริ่มจากของที่มี เลือกเส้นทางอ่านที่ตรงกับคุณ และจำไว้ว่า — เครื่องมือเปลี่ยนเร็ว แต่เหตุผลที่คุณอยากคุมข้อมูลตัวเองจะไม่เปลี่ยน

ก้าวต่อไป

ตอนนี้คุณรู้แล้วว่า "ทำไม" คนถึงเริ่มมอง Local AI และคุณวางความคาดหวังได้ตรงแล้ว — มันเป็นอีกทางเลือก ไม่ใช่ยาวิเศษ

แต่คำว่า Cloud, Local และ Hybrid ยังคลุมเครืออยู่ใช่ไหมครับ มันต่างกันตรงไหนกันแน่ ข้อมูลวิ่งไปไหน ใครจ่ายอะไร และคำว่า "Local AI" กับ "Local LLM" ที่ได้ยินบ่อย ๆ มันเหมือนหรือต่างกัน? บทถัดไปเราจะถอดทั้งสามโมเดลออกมาวางเทียบกันให้เห็นชัด ๆ ทีละมิติ เพื่อให้คุณเริ่มมองออกว่า สถานการณ์แบบไหนเอียงไปทางไหน — เป็นพื้นฐานก่อนที่บทที่ 3 จะให้กรอบตัดสินใจเต็มรูปแบบ