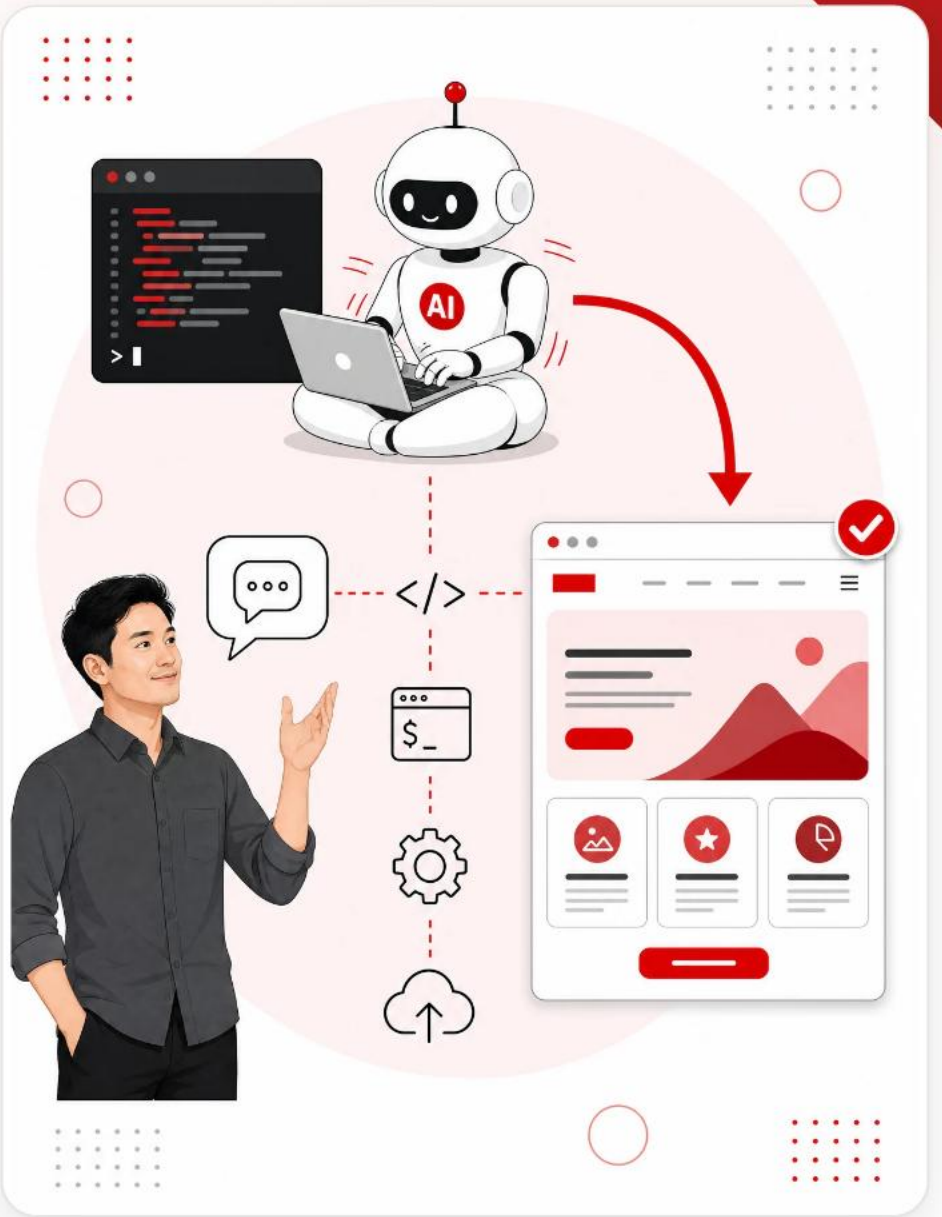
FIRST VIBE CODING FOR BEGINNERS

# First Vibe Coding for Beginners

## ปูพื้นฐาน Vibe Coding
## ใช้ AI ลงมือสร้างงานได้จริง

จากแนวคิด → เข้าใจภาพรวม → ลงเครื่อง → สั่งงานให้เป็น → ลงมือทำจริง
เริ่มจากศูนย์ได้ แม้ไม่มีพื้นฐานโปรแกรมมิ่ง — **โดยสถาบัน ZenityX**



กำหนดการวันนี้

# Timeline 1 วันเต็ม — 10:00 น. – 17:00 น.

## ช่วงเช้า · เรียนรู้ & เตรียม ENVIRONMENT

**10:00–10:30** — **ทำความรู้จักกับการ Vibe Coding**
Vibe Coding คืออะไร ได้อะไร เสียอะไร และบทบาทใหม่ของคุณ

**10:30–11:00** — **Vibe Coding สามารถทำอะไรได้บ้าง**
สำรวจสิ่งที่ AI สร้างได้จริง และศัพท์เทคนิคที่ควรรู้

**11:00–12:30** — **เตรียม Environment ให้พร้อมพัฒนา**
ติดตั้ง Claude Desktop + Git · จัดโฟลเดอร์โปรเจกต์ · เตรียมบัญชี Supabase และ Cloudflare

**12:30–13:30** — **พักกลางวัน**
รับประทานอาหารกลางวัน

## ช่วงบ่าย · หน้าบ้าน → หลังบ้าน → DEPLOY

**13:30–14:30** — **พาทำระบบหน้าบ้าน (Frontend)**
Workshop 1 · สร้างหน้าเว็บขายสินค้า Sales Page ให้สวยและ Responsive

**14:30–16:00** — **พาทำระบบหลังบ้าน (Backend)**
Workshop 2 เชื่อม Supabase เป็นฐานข้อมูล → Workshop 3 หลังบ้าน เพิ่ม · แก้ · ลบสินค้า

**16:00–16:30** — **Deploy เพื่อใช้งานจริง**
Workshop 4 · ขึ้น Cloudflare Pages ให้คนอื่นเข้าใช้งานได้จริง

**16:30–17:00** — **Q&A สำหรับการพัฒนาต่อยอด**
ถาม-ตอบ และแนวทางต่อยอดหลังจบคอร์ส

# 01

# มาทำความรู้จักกับการ Vibe Coding

Vibe Coding คืออะไร · ได้อะไร · เสียอะไร
— และบทบาทใหม่ของคุณในฐานะผู้กำกับ AI

## Vibe Coding เหมาะกับใคร



**01**

**บุคคลทั่วไป**

อยากสร้างเว็บและแอปเอง



**02**

**คนทำงาน · ผู้ประสานงาน**

คุยกับทีมพัฒนาให้รู้เรื่อง




**03**

**ผู้สนใจโลก AI Coding**

ลงมือสร้างผลงานจริง

PART 1 · รู้จัก VIBE CODING

# Vibe Coding คืออะไร เหมาะกับใคร

การเขียนโปรแกรมรูปแบบใหม่ที่ใช้ **AI เขียนโค้ดให้ทั้งหมด** โดยเราแค่พิมพ์สั่งงานด้วยภาษาคน

- **บุคคลทั่วไป** — อยากสร้างเว็บหรือแอปเอง โดยไม่ต้องมีพื้นฐานโค้ด
- **คนทำงาน · ผู้ประสานงาน** — ต้องคุยกับทีมพัฒนาให้รู้เรื่องและเห็นภาพระบบ
- **ผู้สนใจโลก AI Coding** — อยากลงมือสร้างผลงานจริงอย่างเป็นระบบ

> ไม่ต้องจำ syntax — ขอแค่ **สื่อสารกับ AI ให้รู้เรื่อง**

# สิ่งที่ได้มา และสิ่งที่ต้องระวัง

## สิ่งที่ได้มา

- **ทำงานได้รวดเร็ว**
  สร้างต้นแบบได้ในไม่กี่ชั่วโมง
- **ลดกำแพงผู้เริ่มต้น**
  ไม่มีพื้นฐานโค้ดก็สร้างได้
- **โฟกัสที่ไอเดีย**
  ไม่ต้องจำคำสั่ง

## สิ่งที่ต้องระวัง

- **ความปลอดภัยของโค้ด**
  อาจมีช่องโหว่แฝง
- **ยังต้องมีความรู้พื้นฐาน**
  ไม่เข้าใจระบบ ต่อยอดยาก
- **ต้องตรวจทานเสมอ**
  AI เข้าใจโจทย์คลาดเคลื่อนได้

**10x เร็วขึ้นในการสร้างต้นแบบ — แต่ต้องรีวิวก่อนใช้จริง**

เร็วขึ้นจริง แต่ **โค้ดที่ทำงานได้ ≠ โค้ดที่ปลอดภัยและต่อยอดได้** — รีวิวก่อนใช้จริงเสมอ




PART 1 · รู้จัก VIBE CODING

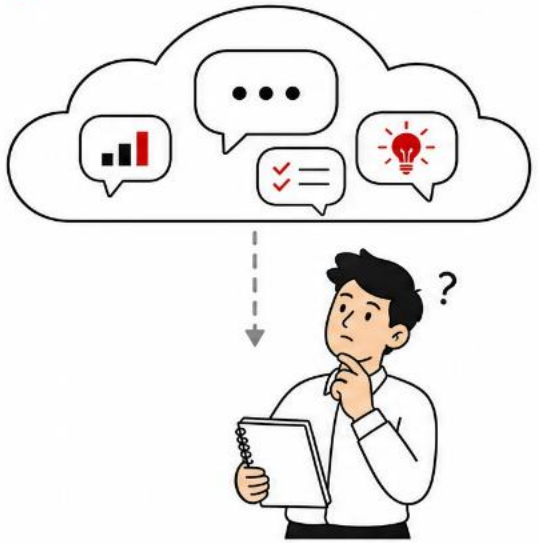
# บทบาทใหม่ของคุณ: AI Director

ไม่ต้องเขียนโค้ดเป็นทุกบรรทัด แต่ต้องเข้าใจภาพรวม — คุณคือคน **สั่ง · ตรวจ · ตัดสินใจ**

- **สั่งงาน · Direct** — บอกความต้องการให้ชัด ทั้งหน้าตา ฟังก์ชัน และสไตล์
- **ตรวจสอบ · Review** — อ่านสิ่งที่ AI เสนอ ดูว่าตรรกะถูกก่อนกดรับ
- **ตัดสินใจ · Approve** — อนุมัติหรือปฏิเสธ คุณคือคนตัดสินใจสุดท้ายเสมอ

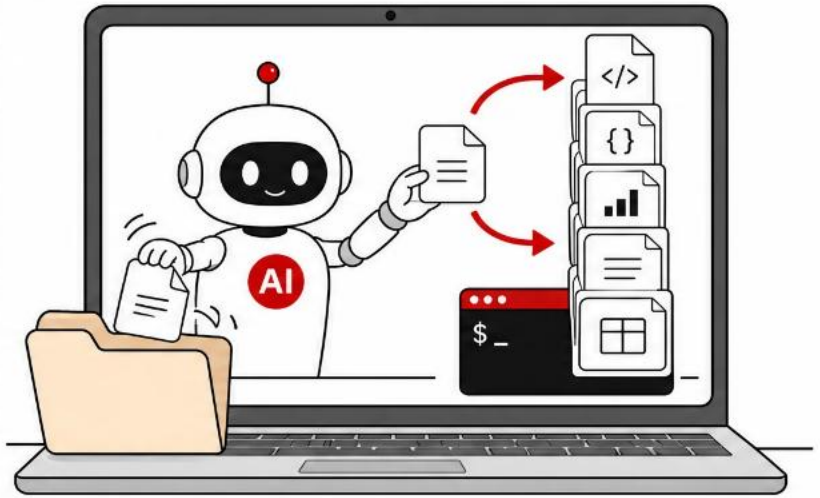
> หัวใจ: ไม่ต้องเก่งโค้ด แต่ต้องเก่ง **“การสื่อสารสิ่งที่ต้องการ”** และ **“การตรวจงาน”**

# ที่ปรึกษา กับ ผู้ลงมือทำ ต่างกันอย่างไร



**Consultant · ที่ปรึกษา**

บอกวิธี แต่คุณต้องลงมือทำเอง




**Executor · ผู้ลงมือทำ**

สร้างไฟล์ รันคำสั่งเองในเครื่องคุณ

**ที่ปรึกษาบอกวิธี — ผู้ลงมือทำ ทำให้จริงในเครื่องของคุณ**

Consultant = ChatGPT · Gemini · Grok | Executor = Claude Code · Codex · Antigravity — **ตัวหลังสิงอยู่ในโฟลเดอร์เครื่องคุณ**

# 02

# Vibe Coding ทำอะไรได้บ้าง

สำรวจสิ่งที่ AI สร้างได้จริง · ยกเว็บแอปเป็นตัวอย่าง
— และศัพท์เทคนิคพื้นฐานที่ควรรู้

# เลือกเครื่องมือให้ถูกกับงาน

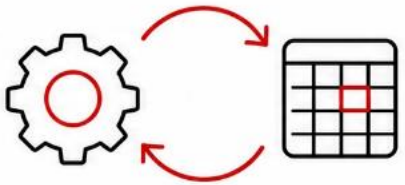


## AI Chat

งานทำครั้งเดียว ใช้ความคิดสร้างสรรค์

สรุปไฟล์ PDF
ร่างอีเมล

เหมาะสุด

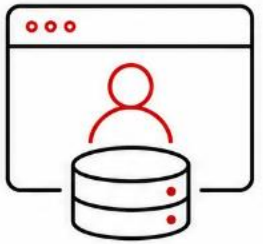


## Automation

งานซ้ำทุกวัน มีกฎตายตัว

เช็คชื่อจาก Excel
สรุปยอดรายวัน

เหมาะสุด



## Web Application

ผู้ใช้ต้องโต้ตอบ และเก็บลงฐานข้อมูล

ระบบเบิกค่าใช้จ่าย
ระบบจองห้อง

เหมาะสุด

**เลือกผิดเครื่องมือ = เสียเวลาเปล่า เลือกถูก = จบงานเร็ว**

3 กลุ่มงานหลักที่ Vibe Coding ทำได้จริง: **AI Chat** ช่วยคิด/เขียน · **Automation** ระบบอัตโนมัติ · **Web Application** เว็บแอปที่ใช้งานได้จริง

# Flow การทำงานของเว็บแอปพลิเคชัน



ผู้ใช้งาน

**Front-End**
หน้าร้าน — สิ่งที่ผู้ใช้เห็น

ส่งคำขอ

**Back-End**
ห้องครัว — ประมวลผลลอจิก

ค้นข้อมูล

**Database**
คลังวัตถุดิบ — เก็บข้อมูล

ส่งข้อมูลกลับไปแสดงผล

ผู้ใช้กด → Front-End → Back-End → Database → ส่งผลกลับมาแสดง

ตัวอย่างหนึ่งที่ได้รับความนิยม: **เว็บแอปพลิเคชัน** — **Front-End** สิ่งที่ผู้ใช้เห็น · **Back-End** สมองที่ตัดสินใจ · **Database** ความจำ

# เว็บแอปพลิเคชัน เหมือนร้านอาหาร




**โซน 1 · หน้าร้าน**
**Front-End**
สิ่งที่ลูกค้าเห็นและสัมผัส

**โซน 2 · ห้องครัว**
**Back-End**
รับออเดอร์ ประมวลผล ปรุงจริง

**โซน 3 · ห้องเก็บของ**
**Database**
เก็บวัตถุดิบอย่างเป็นระเบียบ

**Full-Stack = เจ้าของร้านที่ดูแลได้ครบทั้ง 3 โซน**

เวลาสั่งงาน AI ให้บอกด้วยว่าจะแก้ **โซนไหน** — เช่น “แก้หน้าร้านอย่างเดียว อย่าแตะฐานข้อมูล” จะได้ผลตรงและปลอดภัยกว่ามาก

# ศัพท์ที่เจอบ่อย (1/3) — จากเครื่องเรา สู่ออนไลน์

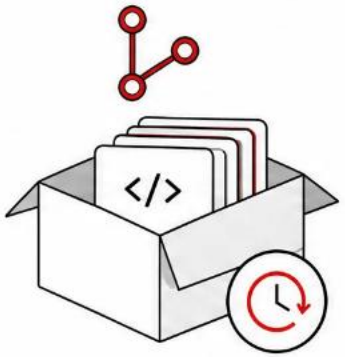


## Localhost

เว็บที่รันบนเครื่องเราเอง

เห็นอยู่คนเดียว ใช้ทดสอบก่อนขึ้นจริง

push ขึ้นคลาวด์



## Repository (Repo)

ที่เก็บโค้ดทั้งโปรเจกต์

อยู่บน GitHub · ย้อนเวอร์ชันเก่าได้

กดปุ่มเดียวขึ้นจริง



## Deploy

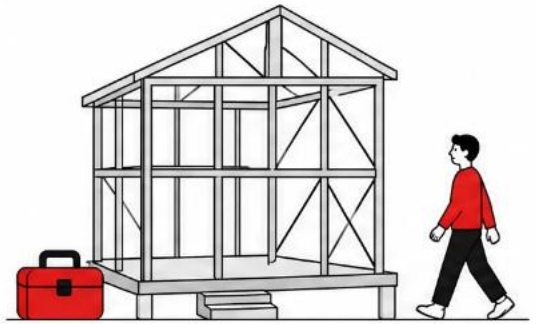
นำเว็บขึ้นออนไลน์

คนทั่วโลกเข้าใช้ได้จริง 24 ชม.

**เขียนที่ Localhost → เก็บที่ Repo → Deploy ให้คนอื่นเข้าได้ — วนแบบนี้ทุกครั้ง**

3 คำนี้คือ **เส้นทางของงานคุณ** — เขียนที่เครื่อง → เก็บบนคลาวด์ → เปิดให้คนอื่นเข้า · Workshop วันนี้จะได้ครบทั้งสามขั้น

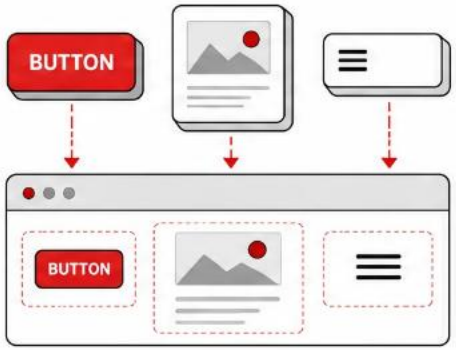
# ศัพท์ที่เจอบ่อย (2/3) — ชิ้นส่วนที่ประกอบเป็นเว็บ



## Framework

โครงสำเร็จรูป

ไม่ต้องเริ่มจากศูนย์ เช่น React




## Component

ชิ้นส่วน UI ที่ใช้ซ้ำได้

ปุ่ม · การ์ด · แถบเมนู ประกอบกันเป็นหน้าเว็บ



## Responsive

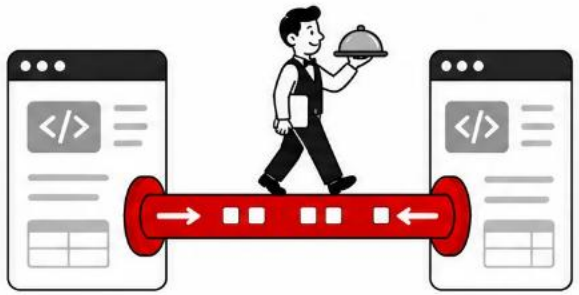
ปรับพอดีทุกจอ

เปิดบนมือถือแล้วยังใช้งานได้สวย

**บอก AI ให้ชัดว่าใช้ Framework อะไร และต้อง Responsive — ได้ผลตรงตั้งแต่ครั้งแรก**

3 คำนี้ใช้ตอน **เขียน prompt** — บอก framework ที่ใช้ · บอกว่าต้องมี component ไหน · บอกว่าต้อง responsive

# ศัพท์ที่เจอบ่อย (3/3) — เชื่อมต่อ และ แก้ปัญหา



## API

**ช่องทางให้โปรแกรมคุยกัน**

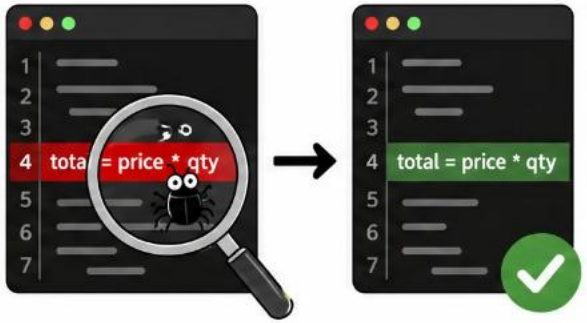
ดึงข้อมูลจากบริการอื่นมาใช้ได้




## API Key

**กุญแจ + บัตรเครดิต ในอันเดียว**

ห้ามแชร์ ห้ามใส่ในโค้ด




## Bug / Debug

**จุดที่ผิด / การแก้จุดนั้น**

เรื่องปกติมาก ส่งให้ AI แก้ได้เลย

**เจอ error ไม่ต้องตกใจ — คัดลอกข้อความสีแดงส่งให้ AI แก้ให้**

**API Key = กุญแจ + บัตรเครดิต** ห้ามหลุดเด็ดขาด · ส่วน **Bug** เป็นเรื่องปกติมาก เจอแล้วส่งให้ AI แก้ได้เลย

PART 2 · VIBE CODING ทำอะไรได้บ้าง

# AI Agent vs. จ้างโปรแกรมเมอร์

ไม่ใช่คู่แข่งกัน — คนละงาน คนละสเกล

- **งานต้นแบบ · งานซ้ำ ๆ · เครื่องมือใช้ในทีม** → Agent คุ้มกว่าหลายสิบเท่า
- **ระบบใหญ่ ซับซ้อน มีคนใช้จริงจำนวนมาก** → ยังต้องมีทีมพัฒนารับผิดชอบ
- ใช้ Agent ทำต้นแบบให้เห็นภาพก่อน แล้วค่อย **ส่งต่อให้ทีมพัฒนาต่อยอด** — ประหยัดทั้งเงินและเวลาสรรหา

> ข้อแลกเปลี่ยนเดียวของ Agent คือ **คุณต้องลงมือคุมเอง** — ซึ่งคือทั้งหมดที่คอร์สนี้สอน



# 03

# เตรียม Environment ก่อนเริ่มพัฒนา

ติดตั้งเครื่องมือ · จัดโฟลเดอร์โปรเจกต์ · เตรียมบัญชี
— และตั้งกติกาก่อนปล่อยให้ Agent ทำงาน

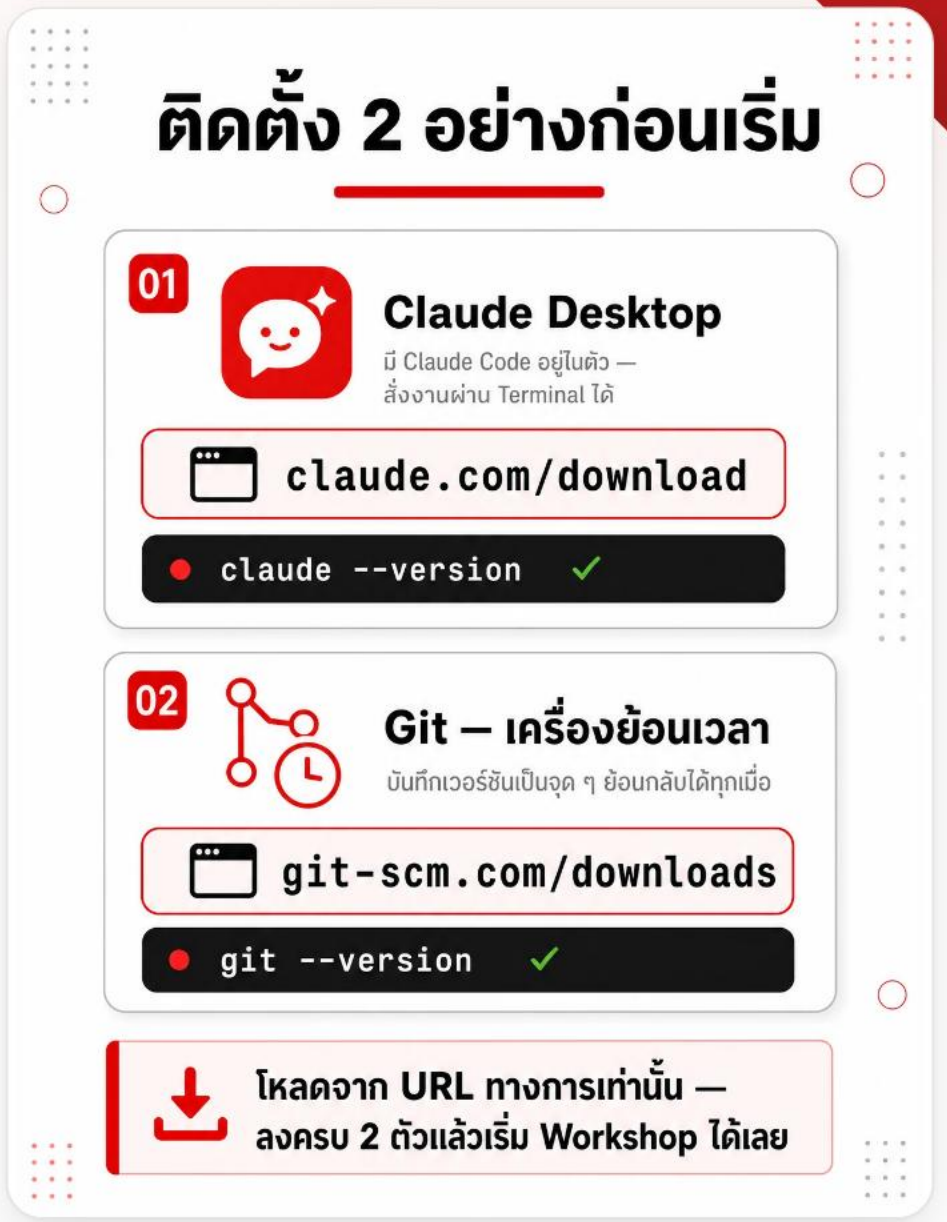
PART 3 · เตรียม ENVIRONMENT

# เครื่องมือที่ต้องติดตั้ง 2 ตัว

- **Claude Desktop** (มี Claude Code ในตัว) — สร้างไฟล์และรันคำสั่งได้เองผ่าน Terminal
  `claude.com/download` · เช็คด้วย `claude --version`
- **Git — เครื่องย้อนเวลา** — บันทึกเวอร์ชันเป็นจุด ๆ (commit = save point) ย้อนกลับได้เสมอ
  `git-scm.com/downloads` · เช็คด้วย `git --version`

> ลงทั้งสองตัวให้เรียบร้อย แล้วเปิด Terminal ในโฟลเดอร์โปรเจกต์ พิมพ์ `claude` เริ่มได้ทันที

ดาวน์โหลด Claude Code → | ดาวน์โหลด Git →






PART 3 · เตรียม ENVIRONMENT

# GitHub ทำอะไรได้บ้าง

ที่เก็บโค้ดออนไลน์ + ระบบจัดการเวอร์ชัน — เหมือน **“Google Drive สำหรับโค้ดที่ย้อนเวลาได้”**

- **สำรองอัตโนมัติ** โค้ดไม่หาย เข้าถึงได้ทุกที่
- **ย้อนเวอร์ชันได้** ดูประวัติทุกการเปลี่ยนแปลง กลับไปจุดที่ยังดีได้
- **ต่อ Deploy อัตโนมัติ** push โค้ดแล้วเว็บอัปเดตเองผ่าน Vercel / Netlify

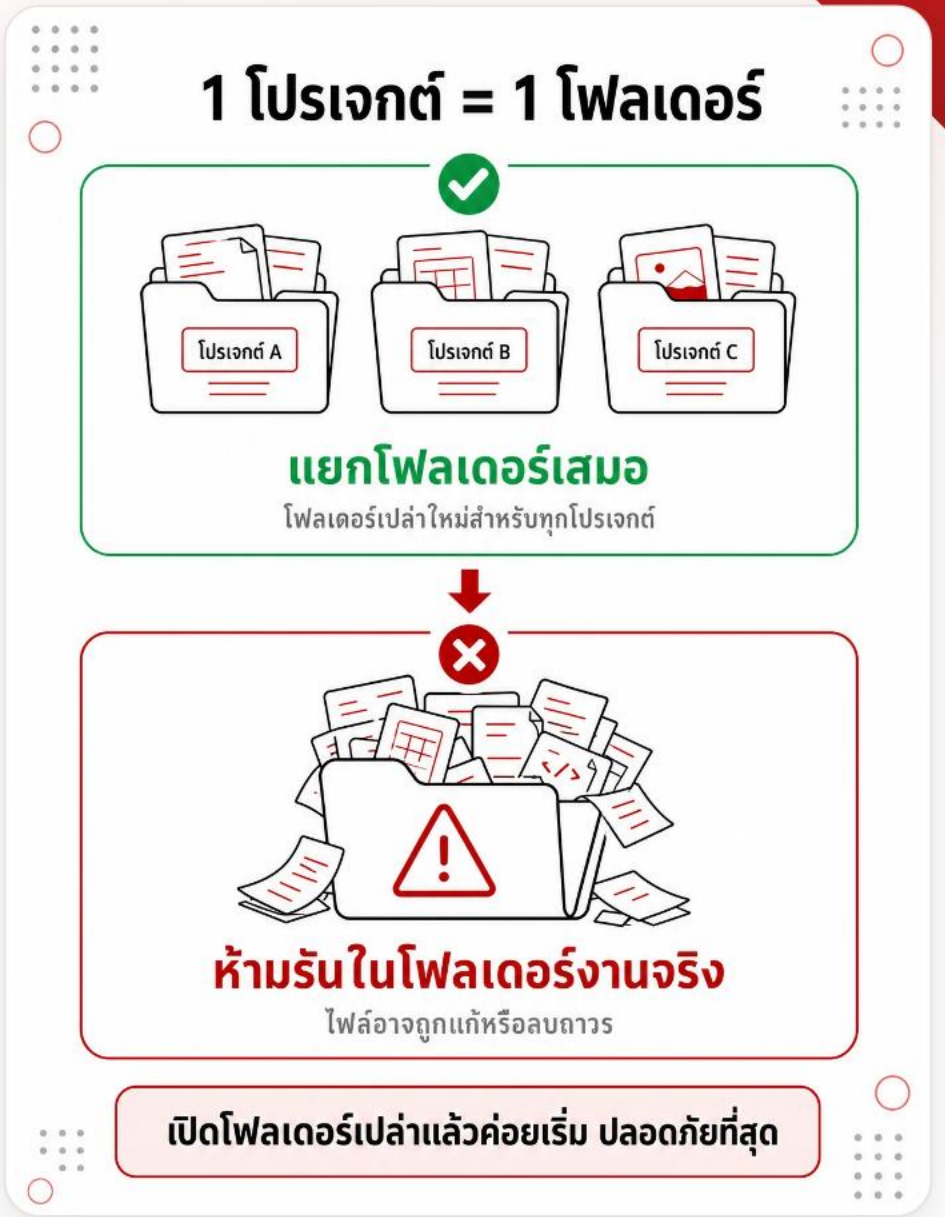
> Vibe Coding ใช้ Git เก็บงานแต่ละขั้น — **AI แก้แล้วพัง ก็ย้อนกลับได้เสมอ**

PART 3 · เตรียม ENVIRONMENT

# โฟลเดอร์ = โต๊ะทำงานของ Agent

- เปิด Agent ที่โฟลเดอร์ไหน **มันเห็นแค่นั้น** — อ่าน แก้ สร้างไฟล์ในขอบเขตนั้น
- **1 โปรเจกต์ = 1 โฟลเดอร์เปล่า** เช่น `~/Projects/company-profile/`
- เปิดกว้างเกิน (ทั้ง Desktop) = สับสน + เสี่ยงไฟล์งานจริงถูกแก้
- ประวัติแชทและการตั้งค่า **ผูกกับโฟลเดอร์** — เปิดที่เดิมคือทำงานต่อได้เลย

> เคล็ดลับ: เปิดโฟลเดอร์เปล่าแล้วสั่ง `claude` — ปลอดภัยและจัดการง่ายที่สุด






PART 3 · เตรียม ENVIRONMENT

# Permission Modes — ระดับการคุม Agent

Claude Code เข้าถึงไฟล์ในเครื่องได้จริง — เลือกระดับอิสระให้เหมาะกับความเสี่ยง (สลับด้วย `Shift+Tab`)

- เริ่มที่ **Manual** เสมอในงานแรก ๆ — เห็นว่ามันจะทำอะไรก่อนกดรับ
- ไว้ใจแล้วค่อยขยับไป **Plan Mode → Accept Edits → Auto**
- **Bypass = ถอดเข็มขัดนิรภัย** ใช้เฉพาะโฟลเดอร์ทดลองที่พังได้

> อ่านก่อนกด yes โดยเฉพาะคำสั่ง **ลบ** และ **เขียนทับ**



# พักกลางวัน

**12:30 – 13:30**

เจอกันอีกครั้ง **13:30 น.**
ช่วงบ่ายเราจะเริ่มลงมือสร้างเว็บขายสินค้าของคุณเอง

PART 3 · เตรียม ENVIRONMENT

# CLAUDE.md ใช้ทำอะไรได้บ้าง

ไฟล์ Markdown ที่รากของโปรเจกต์ — Claude อ่านก่อนทำงาน **ทุกครั้ง** เหมือน “คู่มือพนักงานใหม่” · เขียนครั้งเดียว ไม่ต้องพิมพ์ซ้ำทุกแชท

- **ล็อกสไตล์งาน**
  “โทนขาว-แดง ฟอนต์ Sarabun เท่านั้น” — ไม่ต้องบอกใหม่ทุกครั้ง
- **ล็อกเครื่องมือที่ใช้**
  “HTML ไฟล์เดียว ห้ามลง framework เพิ่ม” — กัน AI พาไปไกลเกินโจทย์
- **ตั้งข้อห้าม**
  “ห้ามลบข้อมูลลูกค้าโดยไม่ถาม” — เซฟตัวเองตั้งแต่ต้น
- **เก็บคำสั่งประจำ + ภาษาที่ตอบ**
  “รันเดฟ npm run dev · ตอบภาษาไทยเสมอ”

พิมพ์ `/init` ให้ Claude ร่างให้อัตโนมัติ แล้วค่อยแก้ — **เขียนสั้น ๆ เป็น bullet** เพราะถูกโหลดทุก session

```
# เว็บขายสินค้า — ร้าน [ชื่อร้าน]
## Tech Stack
- HTML + CSS ไฟล์เดียว เท่านั้น
## Rules
- สีแบรนด์: ขาว + แดง #E60000
- ฟอนต์: หัวข้อ IBM Plex Sans Thai
- รูปสินค้าเก็บในโฟลเดอร์ images/
- ราคาแสดงเป็นบาท มีคอมมา เช่น 1,290
- ทุกหน้าต้อง Responsive
## Commands
- เปิดดู: open index.html
## Don’ts
- ห้ามลบสินค้าออกจากซีตโดยไม่ถามก่อน
- ห้ามเปลี่ยนสี/ฟอนต์เองโดยไม่ถาม
- ตอบภาษาไทยเสมอ
```

PART 3 · เตรียม ENVIRONMENT

# Skill (SKILL.md) คืออะไร

โฟลเดอร์ที่บรรจุ “วิธีทำงานชนิดหนึ่ง” — **เขียนครั้งเดียว ใช้ซ้ำได้ทุกโปรเจกต์**

- **วางที่ `~/.claude/skills/`**
  Claude หยิบไปใช้เองเมื่องานเข้าเงื่อนไข
- **เรียกด้วย `/ชื่อ-skill`**
  หรือปล่อยให้มันเลือกใช้เองอัตโนมัติ
- **ตัวอย่าง: ธีมสไลด์ ZenityX**
  บรีฟ CI ครั้งเดียว ใช้ได้ทุกเด็คหลังจากนั้น

**CLAUDE.md = นิสัยเฉพาะโปรเจกต์ · Skill = ทักษะอาชีพที่ติดตัวข้ามทุกโปรเจกต์**

```
~/.claude/skills/zenityx-slides/
  SKILL.md    ← ความเชี่ยวชาญ
  assets/       ← ไฟล์ประกอบ
  references/   ← เอกสารเพิ่มเติม

# SKILL.md
---
name: zenityx-slides
description: สร้างสไลด์ธีม ZenityX...
---
# ทำสไลด์แบบ ZenityX
(ขั้นตอนที่ Claude จะทำตาม)
```

# 04

# สั่งงานให้เป็น

Prompt ที่ได้ผล · Golden Workflow · Plan Mode
— แก้บั๊ก และคุมค่าใช้จ่าย

PART 4 · สั่งงานให้เป็น

# 4 ขั้นตอน Prompt ให้ได้ผลสูงสุด

1. **บอกเป้าหมาย · What**
   มีส่วนประกอบไหนบ้าง บอกให้ครบทุก section
2. **บอกสไตล์ · Look**
   โทนสี ฟอนต์ อารมณ์ ให้ AI จับทางถูก
3. **ระบุตรรกะ · Logic**
   กดปุ่มแล้วเกิดอะไร ข้อมูลไปไหน แก้ไขได้ไหม
4. **ขอสรุป & ทดสอบ · Review**
   ให้สรุปไฟล์ที่แก้ แล้วเปิดให้ดูผล

```
// ตัวอย่าง prompt ที่ดี

"ทำเว็บขายสินค้าร้าน [ชื่อร้าน]
โครงสร้าง: Hero (ชื่อร้าน+สโลแกน+ปุ่มดูสินค้า),
  รายการสินค้าแบบการ์ด, เกี่ยวกับร้าน, ติดต่อ/สั่งซื้อ
สไตล์: โทนขาว-แดง สะอาด ดูน่าเชื่อถือ
ฟังก์ชัน: การ์ดสินค้ามี รูป ชื่อ ราคา ปุ่มสั่งซื้อ
ตรรกะ: กด 'สั่งซื้อ' → เปิดฟอร์ม (ชื่อ/เบอร์/จำนวน)
  แล้วขึ้นข้อความขอบคุณ
สรุป: บอกไฟล์ที่แก้ + เปิดให้ดูผล Responsive"
```

ถ้าผลลัพธ์ยังไม่โดน **อย่าเริ่มใหม่** — บอกเพิ่มทีละจุด เช่น “ปุ่มสั่งซื้อเล็กไป ขอใหญ่ขึ้น” แล้ว AI จะแก้ต่อยอดให้

# Prompt แย่ กับ Prompt ดี ต่างกันแค่ไหน

## Prompt แย่ — กว้างเกินไป

ทำหน้าเว็บบริษัทให้หน่อย




- ไม่บอกว่าต้องมีส่วนไหน
- ไม่บอกสไตล์และสี
- ต้องสั่งแก้ซ้ำหลายรอบ

## Prompt ดี — ชัดเจน ครบ

ทำ Landing Page โทนขาว-แดง
มี Hero, บริการ 3 การ์ด, ฟอร์มติดต่อ
กดส่งแล้วขึ้นข้อความขอบคุณ



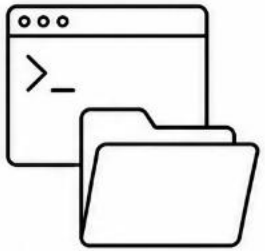

- บอกทุกส่วนที่ต้องมี
- ระบุสไตล์และสีแบรนด์
- บอกตรรกะปุ่มและฟอร์ม

**สูตร Prompt ดี = บริบท + สไตล์ + ตรรกะ + ผลลัพธ์ที่ต้องการ**

เขียน prompt ให้ละเอียดขึ้น 30 วินาที = ประหยัดการสั่งแก้ 5-10 รอบ และ **ประหยัด token** ไปในตัว

# 4 ขั้นตอน สั่งงานฉบับมือโปร

**STEP 01**



## Initialize

เปิดโฟลเดอร์เปล่าที่เตรียมไว้

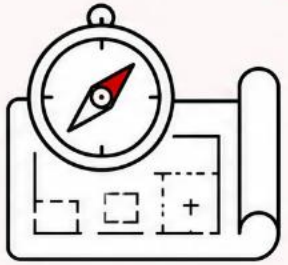
**STEP 02**



## Context

อธิบายเป้าหมายให้ละเอียด

**STEP 03**



## Plan Mode

ให้ AI ร่างโครงสร้างก่อนเขียนโค้ด

**STEP 04**



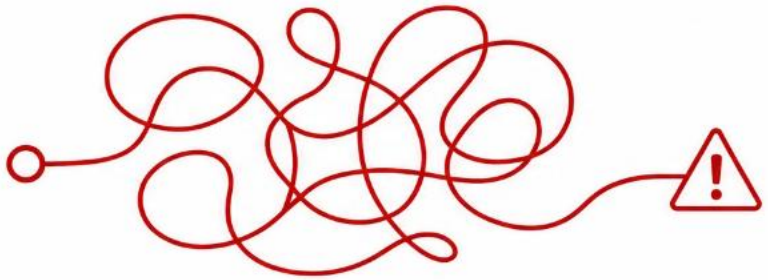
## Accept and Auto

ตรวจแผนแล้วปล่อยให้ระบบสร้าง

**หัวใจอยู่ที่ขั้นที่ 3 — Plan Mode บังคับให้วางแผนก่อนลงมือ**

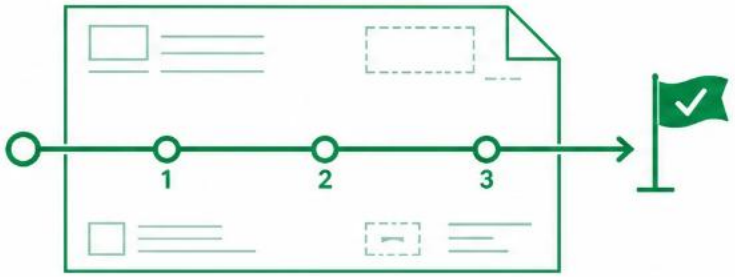
ทำครบ 4 ขั้นทุกครั้งแม้เป็นงานเล็ก — **ความสม่ำเสมอคือสิ่งที่ทำให้ผลลัพธ์นิ่ง** ไม่ใช่ความเก่งของ prompt เดียว

# ทำไม Plan Mode จึงสำคัญที่สุด



## Path A — สั่งปุ๊บ ทำปั๊บ

เริ่มเร็ว แต่ผิดพลาดสูง
วนลูปแก้โค้ดซ้ำไม่จบ เปลืองโควตา




## Path B — วางแผนก่อน

เห็นโครงสร้างไฟล์และ Flow ทั้งหมดก่อน
แก้ทิศทางได้ทันทีก่อนเขียนโค้ดบรรทัดแรก

**วางแผนก่อน 1 นาที ประหยัดการแก้โค้ดเป็นสิบรอบ**

กด `Shift+Tab` เพื่อเข้า Plan Mode — อ่านแผนให้จบ แก้ทิศทางตรงนั้น แล้วค่อยกดอนุมัติ **ตอนนั้นแก้ฟรี ตอนโค้ดเสร็จแล้วแก้แพง**

PART 4 · สั่งงานให้เป็น

# เมื่อแอปเกิดปัญหา — ไม่ต้องตกใจ

เจอบั๊กแล้วแก้ได้ คือ **ทักษะสำคัญที่สุด** ของนักพัฒนา ทุกคนเจอเป็นเรื่องปกติ

// ตัวอย่าง prompt แก้บั๊ก

```
หน้าจอขึ้น error สีแดงตามนี้:
TypeError: cannot read
property 'map' of undefined
ช่วยหาสาเหตุแล้วแก้ให้หน่อย
บอกด้วยว่าแก้ไฟล์ไหนไปบ้าง
```

> คัดลอก error **ทั้งก้อน** หรือแคปหน้าจอวางไปเลย — อย่าพิมพ์สรุปเอง เพราะรายละเอียดที่ตัดทิ้งมักคือจุดที่ผิด






PART 4 · สั่งงานให้เป็น

# บริหาร Token & ค่าใช้จ่าย

ยิ่งบทสนทนายาว ค่าใช้จ่ายยิ่งสะสม — จัดการนิดหน่อยก็ประหยัดได้มาก

- **ขึ้นงานใหม่ = เริ่มแชทใหม่** อย่าลากแชทเดียวข้ามหลายเรื่อง
- **อย่าเทของไม่จำเป็นเข้าไป** วางเฉพาะส่วนที่เกี่ยว ไม่ใช่เอกสารทั้งเล่ม
- **ใส่บริบทถาวรไว้ใน CLAUDE.md** จะได้ไม่ต้องอธิบายซ้ำทุกครั้ง

> Agent วนหลายรอบ (อ่าน → คิด → ทำ → เช็ค) = กิน token **มากกว่าแชทธรรมดาหลายเท่า**

# 05

# มาเริ่มทำ Workshop กันเลย

หน้าบ้าน → หลังบ้าน (Supabase + CRUD) → Deploy
— เว็บขายสินค้าที่จัดการสินค้าเองได้ และขึ้นออนไลน์จริง

PART 5 · WORKSHOP & ขึ้นจริง

# Workshop 1 · ระบบหน้าบ้าน — เว็บ ขายสินค้า ของคุณเอง

ขายอะไรก็ได้ที่คุณอยากขาย — เสื้อผ้า ขนม งานคราฟต์ คอร์สเรียน บริการรับจ้าง

1. **Hero ร้าน**
   ชื่อร้าน + สโลแกน + ปุ่ม "ดูสินค้า"
2. **รายการสินค้า**
   การ์ดสินค้า: รูป + ชื่อ + ราคา + ปุ่มสั่งซื้อ
3. **เกี่ยวกับร้าน**
   เล่าที่มา จุดขาย ทำไมต้องซื้อกับคุณ
4. **ติดต่อ / สั่งซื้อ**
   ฟอร์มสั่งซื้อ + LINE / โทร / อีเมลจริง

// ตัวอย่าง prompt เริ่มต้น — แก้ [ ] เป็นร้านของคุณ

```
ทำเว็บขายสินค้าเป็นไฟล์ index.html ไฟล์เดียว
ร้าน: [ชื่อร้าน] ขาย [สินค้าอะไร]
โครงสร้าง: Hero + รายการสินค้า (การ์ด 3 ใบ
  รูป/ชื่อ/ราคา/ปุ่มสั่งซื้อ) + เกี่ยวกับร้าน + ติดต่อ
สไตล์: [โทนสีที่ชอบ] สะอาด ดูน่าเชื่อถือ
ตรรกะ: กด "สั่งซื้อ" → เปิดฟอร์ม
  (ชื่อ / เบอร์ / จำนวน) แล้วขึ้นข้อความขอบคุณ
สรุป: บอกไฟล์ที่แก้ + เปิดให้ดูผล Responsive
```

ได้หน้าเว็บแล้ว ปรับแต่งให้สวย — **ใส่รูปสินค้าจริง · เปลี่ยนฟอนต์ · เช็คบนมือถือ** แล้วไปต่อระบบหลังบ้าน

PART 5 · WORKSHOP & ขึ้นจริง

# Workshop 2–3 · ระบบหลังบ้าน — Supabase + CRUD

ย้ายสินค้าจากโค้ดไปเก็บในฐานข้อมูล แล้วมีหน้าหลังบ้านให้เจ้าของร้านจัดการสินค้าเองได้

1. **สร้างตารางสินค้าใน Supabase**
   ตาราง products: ชื่อ · ราคา · รูป · รายละเอียด
2. **เชื่อมเว็บกับ Supabase**
   ใช้ Project URL + anon key ให้หน้าบ้านดึงสินค้ามาแสดง
3. **หน้าหลังบ้าน (Admin)**
   เพิ่มสินค้า · แก้ราคา · ลบสินค้า (CRUD)
4. **ทดสอบให้ครบวงจร**
   แก้ราคาที่หลังบ้าน → หน้าเว็บเปลี่ยนตามทันที

```
// ตัวอย่าง prompt — ใช้ต่อจากเว็บใน Workshop 1

เชื่อมเว็บขายสินค้านี้กับ Supabase
ฐานข้อมูล: ตาราง products (name, price,
  image_url, description)
หน้าบ้าน: ดึงสินค้าจากตารางมาแสดงแทนข้อมูลในโค้ด
หลังบ้าน: หน้า admin.html เพิ่ม / แก้ราคา / ลบสินค้า
ความปลอดภัย: ใช้ anon key เท่านั้น
  เก็บค่าในไฟล์ config แยก ห้ามใช้ service_role
สรุป: บอกขั้นตอนที่ต้องทำใน Supabase + ไฟล์ที่แก้
```

แยก **หน้าบ้าน** (ลูกค้าเห็น) กับ **หลังบ้าน** (เจ้าของร้านจัดการ) — ข้อมูลอยู่ที่ **Supabase** ที่เดียว แก้ครั้งเดียวเห็นผลทุกหน้า

PART 5 · WORKSHOP & ขึ้นจริง

# Workshop 4 · Deploy — เอาเว็บขึ้นให้คนเข้าได้

บริการเอาเว็บไปวางบนอินเทอร์เน็ต ให้คนทั่วโลกเข้าได้ 24 ชม. — **เริ่มฟรีได้ทั้ง 3 เจ้า**

### Vercel

- ✓ deploy อัตโนมัติจาก GitHub ง่ายสุด
- ✓ optimize Next.js / React ดีที่สุด
- — ฟรีจำกัด bandwidth

**เหมาะกับ:** เว็บ Next.js / React, เดโม่เร็ว

### Netlify

- ✓ ใช้ง่ายมาก ลากไฟล์วางก็ขึ้นได้เลย
- ✓ ไม่ต้องมี GitHub ก็เริ่มได้
- — โปรเจกต์ใหญ่ที่มี backend ต้องตั้งค่าเพิ่ม

**เหมาะกับ:** Landing page, เว็บ static, มือใหม่

### Cloudflare Pages

- ✓ เร็วสุด เครือข่ายทั่วโลก (CDN)
- ✓ bandwidth ฟรีไม่จำกัด + กัน DDoS
- — เครื่องมือ dev น้อยกว่า ตั้งค่าซับซ้อนกว่านิด

**เหมาะกับ:** เว็บคนเข้าเยอะทั่วโลก เน้นเร็ว+ประหยัด

React/Next.js → **Vercel** · คนเข้าเยอะทั่วโลก → **Cloudflare** · **Workshop วันนี้ใช้ Cloudflare Pages** — ได้ลิงก์ฟรี *.pages.dev และใช้กับ Supabase บน production ได้ทันที

# Q&A

## ขอบคุณครับ

รู้จัก Vibe Coding → Vibe Coding ทำอะไรได้บ้าง → เตรียม Environment
→ สั่งงานให้เป็น → หน้าบ้าน · หลังบ้าน · Deploy ขึ้นจริง

พร้อมตอบทุกคำถามเรื่อง **First Vibe Coding for Beginners** — ลงมือสร้างจริงไปด้วยกัน

> อยากต่อยอดเป็นระบบเต็มรูปแบบ — มีตะกร้าสินค้า ระบบสมาชิก และรับชำระเงินออนไลน์? ไปต่อกับคอร์ส **Vibe Coding for Web Application** ได้เลย

zenityx