

ปั้น AI Agent ด้วย AgentKit

จากแนวคิดสู่แอปพลิเคชัน
ที่ใช้งานได้จริง

อนุชิต ชลธร

สำนักพิมพ์ก๊อปวาง

ปั้น AI Agent ด้วย AgentKit

จากแนวคิดสู่แอปพลิเคชันที่ใช้งานได้จริง

อนุชิต ชโลธร

คำนำสำนักพิมพ์

ในยุคที่เทคโนโลยีปัญญาประดิษฐ์ก้าวหน้าอย่างไม่หยุดนิ่ง มีนวัตกรรมเพียงไม่กี่อย่างที่สามารถสร้างแรงสั่นสะเทือนและความเปลี่ยนแปลงได้เท่ากับการเกิดขึ้นของ **AI Agent**

เมื่อเราได้รู้จักแนวคิดเบื้องหลัง **AgentKit** ของ OpenAI เป็นครั้งแรก เราได้รู้ได้ทันทีว่านี่คือเทคโนโลยีที่จะ “นิยามใหม่” ให้กับขอบเขตของสิ่งที่เป็นไปได้ในโลกของซอฟต์แวร์ มันคือการก้าวกระโดดครั้งสำคัญจากเครื่องมือ AI ที่ตอบสนองคำสั่งอย่างจำกัด ไปสู่ระบบอัตโนมัติที่มีเป้าหมายชัดเจนและสามารถตัดสินใจได้อย่างชาญฉลาด

ด้วยเหตุนี้ เราจึงมีความยินดีอย่างยิ่งที่จะนำเสนอหนังสือ “**ปั้น AI Agent ด้วย AgentKit**” ซึ่งไม่ได้เป็นเพียงคู่มือเชิงเทคนิคเท่านั้น แต่ยังเปรียบเสมือน “แผนที่นำทาง” สำหรับการสร้างแอปพลิเคชันอัจฉริยะยุคใหม่ หนังสือเล่มนี้สะท้อนพันธกิจของเราที่มุ่งส่งเสริมศักยภาพนักพัฒนา ด้วยเนื้อหาที่นำไปใช้ได้จริง มองไกล และร่วมสมัยกับเทคโนโลยีล่าสุด

เราเชื่อว่าหนังสือเล่มนี้จะไม่เพียงสอนคุณว่า “จะใช้ AgentKit อย่างไร” แต่还将จุดประกายให้คุณมองเห็น “บทบาทใหม่ของ AI” ในซอฟต์แวร์ที่คุณสร้างขึ้น

เรามั่นใจว่าคู่มือเล่มนี้จะกลายเป็นทรัพยากรอันทรงคุณค่าสำหรับทุกคนที่ต้องการฝึกฝนทั้งศาสตร์และศิลป์แห่งการพัฒนา **AI Agent**

คำนำผู้เขียน

เรากำลังยืนอยู่ในรุ่งอรุณของยุคใหม่แห่งปัญญาประดิษฐ์ **AI** ไม่ได้เป็นเพียงเครื่องมือในการประมวลผลข้อมูลหรือสร้างข้อความอีกต่อไป แต่กำลังก้าวสู่การเป็น “เอเจนต์” (Agent) – ผู้ช่วยอัจฉริยะที่สามารถเข้าใจเป้าหมาย ตัดสินใจ และลงมือทำงานที่ซับซ้อนได้ด้วยตนเอง

เอเจนต์เหล่านี้กำลังเปลี่ยนแปลงวิธีที่เราสื่อสารและทำงานร่วมกับเทคโนโลยีอย่างลึกซึ้ง ตั้งแต่การทำงานอัตโนมัติ การสร้างประสบการณ์ผู้ใช้ที่ตอบโต้เฉพาะบุคคล ไปจนถึงการเปิดประตูสู่แนวทางการแก้ปัญหาใหม่ๆ ที่เราเพิ่งเริ่มค้นพบ

หนังสือ **“ปั้น AI Agent ด้วย AgentKit”** เล่มนี้ออกแบบมาเพื่อพาคุณก้าวเข้าสู่โลกยุคใหม่ของเอเจนต์อย่างมั่นใจ โดยมีเป้าหมายในการถ่ายทอดความรู้และมอบเครื่องมือให้แก่นักพัฒนาทุกระดับ เพื่อสร้าง **AI Agent** ที่มีความซับซ้อนและพร้อมใช้งานจริง ด้วยศักยภาพของ **AgentKit** จาก OpenAI

ตลอดทั้งเล่ม คุณจะได้เรียนรู้จากพื้นฐานของแนวคิด Agentic AI ไปจนถึงการลงมือปฏิบัติจริงกับเครื่องมือสำคัญต่างๆ เช่น **Agent Builder**, **ChatKit** และ **Agents SDK** รวมถึงหัวข้อที่มักถูกมองข้าม

เช่น ความปลอดภัย การประเมินประสิทธิภาพ และการออกแบบเอเจนต์อย่างมีความรับผิดชอบ

ไม่ว่าคุณจะเป็นนักพัฒนามากประสบการณ์ที่ต้องการยกระดับผลิตภัณฑ์ของคุณด้วยเอเจนต์อัจฉริยะ หรือเป็นผู้เริ่มต้นที่อยากเข้าใจอนาคตของ AI หนังสือเล่มนี้ถูกสร้างขึ้นเพื่อคุณโดยเฉพาะ ด้วยเนื้อหาที่ไล่เรียงจากพื้นฐานไปสู่เทคนิคขั้นสูง พร้อมตัวอย่างที่สามารถนำไปใช้ได้จริง

ยุคของ AI Agent มาถึงแล้ว และนี่คือเวลาของคุณที่จะร่วมสร้างอนาคตไปด้วยกัน

สารบัญ

บทที่ 1: ความรู้เบื้องต้นเกี่ยวกับ Agents และ AgentKit	1
Agents คืออะไร?	1
แนะนำ AgentKit	2
วิธีการสร้างเอเจนต์	2
การนำเอเจนต์ไปใช้งาน	4
การเพิ่มประสิทธิภาพเอเจนต์	5
บทที่ 2: แนะนำ Agent Builder	8
แนะนำ Agents และ Workflows	9
การประกอบ Workflow	10
เวิร์กโฟลว์และเทมเพลต	11
ประเภทโหนด	13
การทดสอบเวิร์กโฟลว์และดีบั๊ก	14
ความปลอดภัยและความเสี่ยง	15
การประเมินคุณภาพ	15
การเผยแพร่เวิร์กโฟลว์	16
การใช้งานร่วมกับแอปพลิเคชัน	18
บทที่ 3: ทำความรู้จักกับ Nodes ใน Agent Builder	20
โหนดหลัก (Core Nodes)	20

โนหนด Start	21
โนหนด Agent	22
โนหนด Note	24
โนหนดเครื่องมือ (Tool Nodes)	24
File Search	25
Guardrails	26
MCP (Model Context Protocol)	27
โนหนดตรรกะ (Logic Nodes)	29
If/Else	30
While	31
User Approval	32
โนหนดข้อมูล (Data Nodes)	33
Transform	34
Set State	35
บทที่ 4: สร้างเกราะป้องกันให้ Agent	37
ทำความเข้าใจสนามรบ: ประเภทความเสี่ยงที่ต้องรู้	37
การโจมตีที่เปลี่ยนทิศทางคำสั่งของ AI (Prompt Injection)	37
การรั่วไหลของข้อมูลลับโดยไม่ตั้งใจ (Private Data Leakage)	38
ชุดเครื่องมือป้องกัน: กลยุทธ์สร้างความปลอดภัยหลายชั้น	38
กลยุทธ์ที่ 1: ควบคุมการไหลของข้อมูลให้รัดกุม	38

กลยุทธ์ที่ 2: ชี้แนะพฤติกรรมของเอเจนต์ให้ชัดเจน	39
กลยุทธ์ที่ 3: ให้นักขุขยมีส่วนร่วมในการตัดสินใจ	40
กลยุทธ์ที่ 4: ตรวจสอบและปรับปรุงอย่างต่อเนื่อง	40
ปลอดภัยได้ด้วยการป้องกันหลายชั้น	41
บทที่ 5: สร้างเอเจนต์ด้วย Agent Builder	42
ตัวอย่าง: เอเจนต์ค้นหาหนังสือจากเว็บไซต์	42
สร้างเอเจนต์ด้วย Agent Builder	42
กำหนดโครงสร้างผลลัพธ์เป็น JSON	50
การแสดงผล Widget (ChatKit Widget)	53
การนำเวิร์กโฟลว์ไปใช้งาน	59
ตัวอย่าง: เอเจนต์ตอบคำถามจากข้อมูล (RAG Agent)	66
สร้างเวิร์กโฟลว์และตั้ง Instruction	66
กำหนดเครื่องมือ File Search และ Vector Store	67
ทดสอบเวิร์กโฟลว์	75
ตัวอย่าง: การใช้เอเจนต์หลายตัว (Multi-Agent Workflow)	78
ตัวอย่าง: การเชื่อมต่อ Agent กับ Model Context Protocol	86
แนะนำ MCP Toolbox for Databases	86
สร้าง MCP Server	94
รัน MCP Toolbox Server	102
ทดลองใช้งาน MCP Server	102

การใช้งาน MCP Toolbox for Databases กับ Agent Builder	104
บทที่ 6: การสร้างหน้าตาแชทให้เอเจนต์ด้วย ChatKit	117
แนวทางในการใช้งาน ChatKit	117
ตัวอย่าง: การฝัง ChatKit ลงในแอปพลิเคชัน	119
สร้างโปรเจกต์	119
เตรียม Workflow ID ให้พร้อม	119
สร้าง Backend Endpoint สำหรับยืนยันตัวตน	120
ติดตั้ง ChatKit ในฝั่ง Frontend	131
บทที่ 7: ปรับแต่งหน้าตาเอเจนต์ของคุณ	136
การปรับแต่งหน้าตาแชท	136
การปรับแต่งธีม (Theming)	137
การปรับแต่งหน้าจอเริ่มต้น (Start Screen)	138
การเพิ่มฟังก์ชันการทำงาน	139
การใช้งาน @Mentions (Entity Tags)	141
การตั้งค่าอื่นๆ	142
การปรับแต่งด้วย ChatKit Playground	143
บทที่ 8: สื่อสารกับผู้ใช้ให้เหนือกว่าแค่ข้อความด้วย Widgets	146
วิดเจ็ตคืออะไร?	146
เริ่มต้นง่ายๆ ด้วย Widget Builder	146
โครงสร้างวิดเจ็ต	149

คอมโพเนนต์ที่สำคัญ	149
คอมโพเนนต์สำหรับจัด Layout	149
คอมโพเนนต์สำหรับแสดงผลข้อมูล	150
คอมโพเนนต์สำหรับโต้ตอบกับผู้ใช้	150
ตัวอย่างการประกอบวิดเจ็ต	150
การจัดการเมื่อผู้ใช้คลิก onAction Callback	152
บทที่ 9: ทำให้วิดเจ็ตมีชีวิตด้วย Actions	154
Action คืออะไร?	154
วิธีสร้างและกระตุ้น (Trigger) Action	155
แบบ Declarative – ประกาศไว้ในวิดเจ็ตโดยตรง	155
แบบ Imperative – สั่งยิง Action จากโค้ด Frontend	155
การจัดการ Action ด้วย Client หรือ Server	156
การจัดการที่ฝั่ง Client (เหมาะกับงาน UI)	156
การจัดการที่ฝั่ง Server (ค่าเริ่มต้น)	157
ตัวอย่างการสร้างฟอร์มแบบโต้ตอบ	158
บทที่ 10: การใช้งาน ChatKit Server บนโครงสร้างพื้นฐานของคุณ	161
ตัวอย่าง: การใช้งาน ChatKit Server	161
สร้าง ChatKit Server สำหรับ Backend	162
สร้างหน้าเว็บสำหรับ Frontend	172
บทที่ 11: การประเมินผลการทำงานของเอเจนต์	186

ทำไมการประเมินผลจึงสำคัญอย่างยิ่ง?	186
ชุดเครื่องมือประเมินผลของ OpenAI	187
Datasets: รากฐานของข้อสอบ	187
Evals: เครื่องตรวจข้อสอบอัตโนมัติ	188
Trace Grading: การวิเคราะห์เชิงลึก	188
Prompt Optimizer: ตัวเตอรส์่วนตัวสำหรับ Prompt	188
วงจรการพัฒนาต่อเนื่อง (The Evaluation Flywheel)	188
บทที่ 12: การประเมินด้วย Trace Grading	191
การใช้งาน Trace	192
การใช้ Grader เพื่อประเมินประสิทธิภาพ	194
ปรับปรุง Prompt	196
บทที่ 13: Agents SDK ชุดเครื่องมือสำหรับนักพัฒนา	200
แนะนำ Agents SDK	200
ทำไมควรใช้ Agents SDK	202
คุณสมบัติเด่นของ Agents SDK	202
เริ่มต้นใช้งาน Agents SDK	203
สร้างโปรเจกต์ใหม่	203
ตั้งค่า Virtual Environment	203
เปิดใช้งาน Virtual Environment	204
ติดตั้ง Agents SDK	204
ตั้งค่าตัวแปรสภาพแวดล้อม	204

ตัวอย่างการใช้งานเบื้องต้น	204
การกำหนดค่าเริ่มต้นให้กับ Agent	207
พารามิเตอร์หลักของ Agent	207
ตัวอย่าง Agent พื้นฐาน	207
การใช้ Context สำหรับ Dependency Injection	208
ตัวอย่าง Context ของผู้ใช้	208
การกำหนดประเภทของผลลัพธ์	210
ตัวอย่าง ผลลัพธ์แบบ Structured Output	210
การปรับแต่งพฤติกรรมของ Agent	211
ควบคุมการใช้เครื่องมือของ LLM	212
ควบคุมพฤติกรรมหลังการเรียกใช้เครื่องมือ	213
คำสั่งแบบไดนามิก	215
การโคลน Agent	216
การรัน Agent และลูปรการทำงานของ Agent	217
ตัวเลือกในการรัน Agent ด้วย Runner	217
เบื้องหลังการทำงาน: ลูปรดำเนินการของ Agent	218
การสตรีม	220
การกำหนดค่าการรันส่วนกลาง	220
การจัดการบทสนทนา	221
การจัดการฟีดปลาด	223
การจัดการผลลัพธ์ สตรีมมิ่ง และหน่วยความจำ	225

การจัดการผลลัพธ์	225
การสตรึมมิ่ง	226
จัดการหน่วยความจำระยะยาวของ Agent ด้วย Sessions	229
การเลือกสมองให้ Agent	234
โมเดลจาก OpenAI	234
การใช้โมเดลที่ไม่ใช่ของ OpenAI	236
วิธีการใช้งานในรูปแบบอื่น	237
การใช้โมเดลที่แตกต่างกันสำหรับงานที่ต่างกัน	243
ข้อควรระวังและปัญหาที่พบบ่อย	245
เสริมพลังให้ Agent ด้วยเครื่องมือ	248
Hosted Tools	248
Function Tools	250
สรุปหลักการ	252
การคืนค่าเป็นไฟล์หรือรูปภาพ	255
การสร้าง Function Tool เอง	255
การจัดการข้อผิดพลาด	257
Agents as Tools	258
ปรับแต่ง Agent Tool	260
การปรับแต่งผลลัพธ์จาก Sub-Agent	261
เปิด/ปิดเครื่องมือแบบมีเงื่อนไข	262

การเชื่อมต่อสู่โลกภายนอกด้วย Model Context Protocol (MCP)	265
การเลือกใช้งาน MCP Integration	266
Hosted MCP Tools	266
Streamable HTTP Servers	270
SSE MCP Servers	272
stdio Servers	273
การกรอกร่องมือ	274
การใช้ Prompts	274
การจัดการแคช	275
การสร้างทีมเอเจนต์และการจัดการการทำงานร่วมกันของเอเจนต์	276
การควบคุมการทำงานร่วมกันด้วย LLM (LLM-led Orchestration)	276
การควบคุมการทำงานร่วมกันโดยโค้ด (Code-led Orchestration)	284
เครื่องมือสำหรับนักพัฒนา	286
การทดสอบ Agent อย่างรวดเร็วด้วย REPL	286
การแสดงผลภาพโครงสร้าง Agent	287
บทที่ 14: สร้างเอเจนต์ด้วย Agents SDK	292
ตัวอย่าง: สร้าง Agent API ด้วย FastAPI	292

สร้างโปรเจกต์	293
เปิดใช้งาน Virtual Environment	293
ตั้งค่า Environment Variables	293
ติดตั้ง Dependencies ที่จำเป็น	294
เขียนโค้ด Agent เบื้องต้นด้วย Agents SDK	294
เขียน API ครอบ Agent ด้วย FastAPI	298
รันเซิร์ฟเวอร์ FastAPI	301
ทดสอบ API	302
ทดสอบผ่าน OpenAPI Docs	304
ตัวอย่าง: สร้าง LINE ChatBot เชื่อมต่อกับ Agents SDK	306
ภาพรวมของระบบ	306
สร้างโปรเจกต์ใหม่	307
ติดตั้ง Dependencies	307
ตั้งค่า Environment Variables	308
ดาวน์โหลดโค้ด Agent จาก Agent Builder	308
แก้โค้ด Agent เพื่อรองรับ Session	309
เขียน Webhook Server ด้วย FastAPI	312
รันเซิร์ฟเวอร์ Webhook	315
ทดสอบผ่าน ngrok	315
ทดสอบการทำงานของ ChatBot	316

ตัวอย่าง: การเชื่อมต่อกับ Flutter เพื่อสตรีมข้อความจากเอเจนต์แบบเรียลไทม์	319
สร้างส่วน Backend	319
สร้างโปรเจกต์ใหม่	319
ตั้งค่า Virtual Environment	319
เรียกใช้ Virtual Environment	319
ติดตั้ง Dependencies	320
ตั้งค่า Environment Variables	320
เขียนโค้ดส่วน API	320
รันเซิร์ฟเวอร์	323
สร้างส่วน Frontend	323
สร้างโปรเจกต์ใหม่	324
ติดตั้ง dependency ที่จำเป็น	324
สร้างหน้าโต้ตอบหลัก	324
รันแอปและทดลองใช้	328
ตัวอย่างการใช้งาน ChatKit, Agent SDK, MCP Server และ OpenAPI ร่วมกัน	331
สร้างเซิร์ฟเวอร์ OpenAPI	331
สร้าง MCP Server	341
สร้าง ChatKit Server	342
สร้าง Frontend และฝัง ChatKit	349

ทดสอบ	366
บทส่งท้าย	370
ดาวนั้โหลดซอร์สโค้ด	371

บทที่ 1: ความรู้เบื้องต้นเกี่ยวกับ Agents และ AgentKit

ในยุคที่ปัญญาประดิษฐ์ไม่ได้เป็นเพียง “ผู้ช่วยตอบคำถาม” อีกต่อไป แต่ได้พัฒนาไปสู่ระบบที่สามารถ **คิด วางแผน และลงมือทำงาน แทนมนุษย์ได้จริง**

“เอเจนต์” จึงกลายเป็นรากฐานของซอฟต์แวร์ยุคใหม่ ที่สามารถทำงานแบบอัตโนมัติได้ตามเป้าหมายที่ซับซ้อน โดยอาศัยความเข้าใจบริบท การใช้เหตุผล และความสามารถในการเชื่อมต่อกับเครื่องมือต่างๆ รอบตัว ทั้งหมดนี้กำลังถูกยกระดับให้เป็นจริงได้ง่ายขึ้นด้วย **AgentKit** ชุดเครื่องมือจาก OpenAI ที่ออกแบบมาเพื่อให้นักพัฒนาสามารถสร้าง ทดสอบ และนำเอเจนต์ไปใช้งานได้อย่างครบวงจร

บทนี้จะพาคุณทำความเข้าใจพื้นฐานของเอเจนต์ แนวคิดเบื้องหลัง AgentKit และโครงสร้างหลักของระบบ ตั้งแต่การสร้างเวิร์กโฟลว์ด้วย Agent Builder ไปจนถึงการนำไปใช้งานและการปรับแต่งประสิทธิภาพ เพื่อปูทางสู่การสร้างเอเจนต์ของคุณเอง

Agents คืออะไร?

เอเจนต์ (Agents) คือระบบที่สามารถดำเนินงานที่ได้รับมอบหมายได้อย่างชาญฉลาด ตั้งแต่ภารกิจที่เรียบง่าย ไปจนถึงเวิร์กโฟลว์ที่ซับซ้อน

ซ้อนและมีจุดสิ้นสุดเปิด (open-ended)

OpenAI ได้เตรียมทั้ง โมเดลที่ออกแบบมาเฉพาะสำหรับงานเชิงเอเจนต์ และ เครื่องมือครบชุด สำหรับการสร้าง จัดการ และติดตามประสิทธิภาพของเอเจนต์ ซึ่งทั้งหมดนี้ช่วยให้นักพัฒนาสามารถสร้างระบบอัตโนมัติที่ตอบโต้ได้อย่างแท้จริง

แนะนำ AgentKit

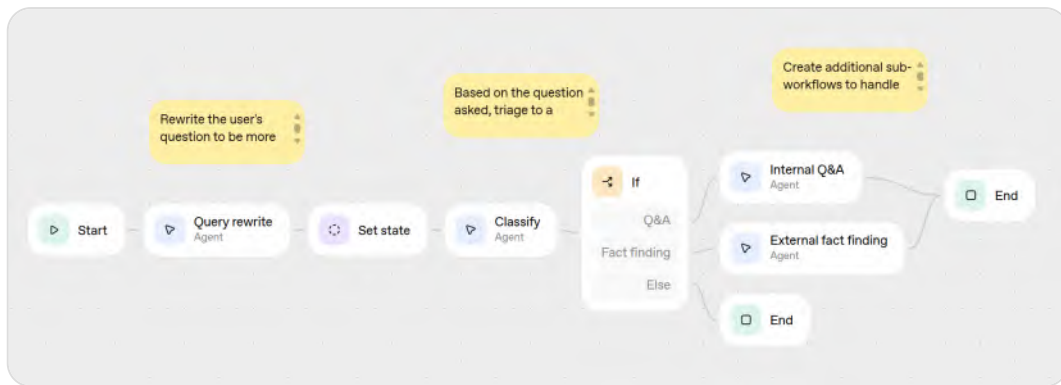
AgentKit คือชุดเครื่องมือแบบโมดูลาร์ (Modular Toolkit) ที่ออกแบบมาเพื่อสนับสนุนการทำงานครบวงจรของเอเจนต์ ตั้งแต่การสร้าง การนำไปใช้งาน จนถึงการเพิ่มประสิทธิภาพ

AgentKit ประกอบด้วย 3 องค์ประกอบหลัก ได้แก่

1. **การสร้าง (Build):** สร้างเวิร์กโฟลว์ด้วย **Agent Builder** ซึ่งเป็น Canvas แบบภาพที่ช่วยให้คุณออกแบบเอเจนต์ได้อย่างเป็นระบบ พร้อมโหมดและเทมเพลตเริ่มต้น
2. **การนำไปใช้ (Deploy):** ใช้ **ChatKit** เพื่อนำเวิร์กโฟลว์ของเอเจนต์ไปฝังในส่วน Frontend ของแอปพลิเคชันได้อย่างรวดเร็ว
3. **การเพิ่มประสิทธิภาพ (Optimize):** สร้างการประเมิน (Evals) เพื่อสังเกต วิเคราะห์ และปรับปรุงการทำงานของเอเจนต์ให้ดียิ่งขึ้น

วิธีการสร้างเอเจนต์

การสร้างเอเจนต์คือกระบวนการออกแบบเวิร์กโฟลว์และเชื่อมโยงส่วนต่างๆ ของแพลตฟอร์ม OpenAI เข้าด้วยกัน **Agent Builder** ได้รวบรวมองค์ประกอบสำคัญทั้งหมดไว้ใน UI เดียว



เวิร์กโฟลว์เอเจนต์

องค์ประกอบหลักที่คุณจะได้ใช้งาน ได้แก่

- **Agent Builder:** เครื่องมือหลักสำหรับสร้างเวิร์กโฟลว์ของเอเจนต์ มีลักษณะเป็น Canvas แบบแผนภาพ ให้คุณรวมโมเดล เครื่องมือ องค์ความรู้ และตรรกะไว้ในที่เดียว
- **OpenAI Models:** เปรียบเสมือน “สมอง” ของเอเจนต์ ที่มีความสามารถในการให้เหตุผล ตัดสินใจ และประมวลผลข้อมูล คุณสามารถเลือกโมเดลที่เหมาะสมกับงานได้โดยตรงใน Agent Builder
- **Tools และ Guardrails:** เพิ่มความสามารถให้เอเจนต์เข้าถึงบริการภายนอกผ่าน MCP ค้นหาข้อมูลใน Vector Stores และป้องกันการใช้งานผิดวัตถุประสงค์ด้วย Guardrails ที่มีมาให้

- **Knowledge และ Memory:** เพิ่มองค์ความรู้ภายนอกให้เอเจนต์ โดยใช้ Vector Stores, File Search และ Embeddings ทำให้เอเจนต์เข้าใจบริบทเฉพาะของคุณมากขึ้น ข้อมูลทั้งหมดถูกโฮสต์โดย OpenAI
- **Logic Nodes:** ควบคุม Control Flow ของระบบ ระบุเงื่อนไขการทำงานร่วมกันของเอเจนต์ และเส้นทางการทำงานที่ซับซ้อนได้
- **Agents SDK:** สำหรับผู้ที่ต้องการความยืดหยุ่นสูงสุด สามารถสร้างเอเจนต์ด้วยการเขียนโค้ด ควบคุมการทำงาน เครื่องมือ และการ Orchestration ได้อย่างเต็มรูปแบบ

การนำเอเจนต์ไปใช้งาน

เมื่อเอเจนต์พร้อมใช้งานแล้ว คุณสามารถนำไปเชื่อมต่อกับแอปพลิเคชันของคุณได้ผ่าน **ChatKit** ซึ่งเป็น UI Component ที่ออกแบบมาให้ฝังใน Frontend ได้ง่าย พร้อมเชื่อมต่อกับ Backend ของคุณโดยตรง

มี 2 แนวทางหลักในการนำเอเจนต์ไปใช้งาน

1. **ChatKit:** คอมโพเนนต์ UI ที่สามารถปรับแต่งได้ ช่วยให้ท่านฝังเวิร์กโฟลว์ของเอเจนต์ในผลิตภัณฑ์ของคุณได้ทันที เพียงระบุ Workflow ID

2. **Advanced ChatKit:** สำหรับการปรับแต่งขั้นสูง คุณสามารถรัน ChatKit บนโครงสร้างพื้นฐานของคุณเอง ผสานกับ widgets และเชื่อมต่อกับ Backend ใดๆ ผ่าน SDKs ได้อย่างอิสระ

การเพิ่มประสิทธิภาพเอเจนต์

หลังจากเอเจนต์ของคุณทำงานได้ตามที่ตั้งใจแล้ว ขั้นตอนถัดมาคือการ **ประเมินและปรับปรุงประสิทธิภาพ** เพื่อให้ตอบสนองได้อย่างแม่นยำ รวดเร็ว และมีคุณภาพสูงสุด

แพลตฟอร์ม **OpenAI** มีเครื่องมือสำหรับการปรับแต่งและประเมินเอเจนต์อย่างเป็นระบบ ได้แก่

- **Evals** คือแพลตฟอร์มสำหรับการประเมินผลแบบครบวงจร ที่ออกแบบมาเพื่อทดสอบและวัดประสิทธิภาพของเอเจนต์ในหลายมิติ ไม่ว่าจะเป็นความถูกต้อง ความสมบูรณ์ หรือความสอดคล้องกับบริบทการใช้งานจริง ผู้พัฒนาสามารถสร้างกรณีทดสอบเฉพาะทาง ประเมินผลลัพธ์ทั้งแบบอัตโนมัติและด้วยมนุษย์ รวมถึงเปรียบเทียบโมเดลหรือการตั้งค่าต่างๆ เพื่อหาค่าที่เหมาะสมที่สุดได้อย่างยืดหยุ่น นอกจากนี้ Evals ยังรองรับการประเมินโมเดลภายนอก ทำให้สามารถเปรียบเทียบผลลัพธ์จากหลายระบบได้อย่างมีประสิทธิภาพ

- **Trace grading** เป็นเครื่องมือที่ช่วยตรวจสอบและให้คะแนน “เส้นทางการทำงาน” (execution trace) ของเอเจนต์ในแต่ละขั้นตอน เพื่อให้เห็นภาพรวมของกระบวนการตัดสินใจ การเรียกใช้เครื่องมือ และการตอบสนองของระบบ ซึ่งมีประโยชน์อย่างมาก ในระหว่างการพัฒนาและการดีบั๊ก นักพัฒนาสามารถใช้ Trace grading เพื่อวิเคราะห์การตัดสินใจที่คลาดเคลื่อน ประเมินคุณภาพผลลัพธ์โดยอัตโนมัติ และปรับปรุง prompt หรือ logic ให้มีประสิทธิภาพมากยิ่งขึ้น
- **Datasets** คือเครื่องมือสำหรับสร้าง จัดการ และติดตามชุดข้อมูลที่ใช้ในการทดสอบเอเจนต์ โดยสามารถจำลองสภาพแวดล้อมการใช้งานจริงและใช้ข้อมูลเดียวกันในการประเมินเอเจนต์หลายรุ่นได้อย่างเป็นระบบ ผู้ใช้สามารถสร้าง dataset สำหรับการทดสอบเฉพาะทาง เช่น คำถาม-คำตอบ บทสนทนา หรือการแก้ปัญหาทางตรรกะ นำมาใช้ร่วมกับ Evals เพื่อวิเคราะห์ผลลัพธ์เชิงลึก และจัดการเวอร์ชันของข้อมูลให้การทดสอบมีความสม่ำเสมอ
- **Prompt optimizer** ทำหน้าที่วิเคราะห์และปรับแต่ง prompt ให้เหมาะสมที่สุด โดยประเมินคุณภาพของผลลัพธ์ ระบุจุดที่ควรปรับปรุง และเสนอแนวทางการเขียนใหม่ เพื่อให้เอเจนต์ตอบสนองได้อย่างแม่นยำและมีประสิทธิภาพ การใช้ Prompt optimizer ช่วยให้ผู้พัฒนาสามารถปรับแต่ง prompt ให้เข้าใจ

ง่าย ลดความกำกวมของเอเจนต์ และเพิ่มความสอดคล้อง
ระหว่างเจตนาของผู้ใช้กับผลลัพธ์ที่ได้อย่างมีประสิทธิภาพ

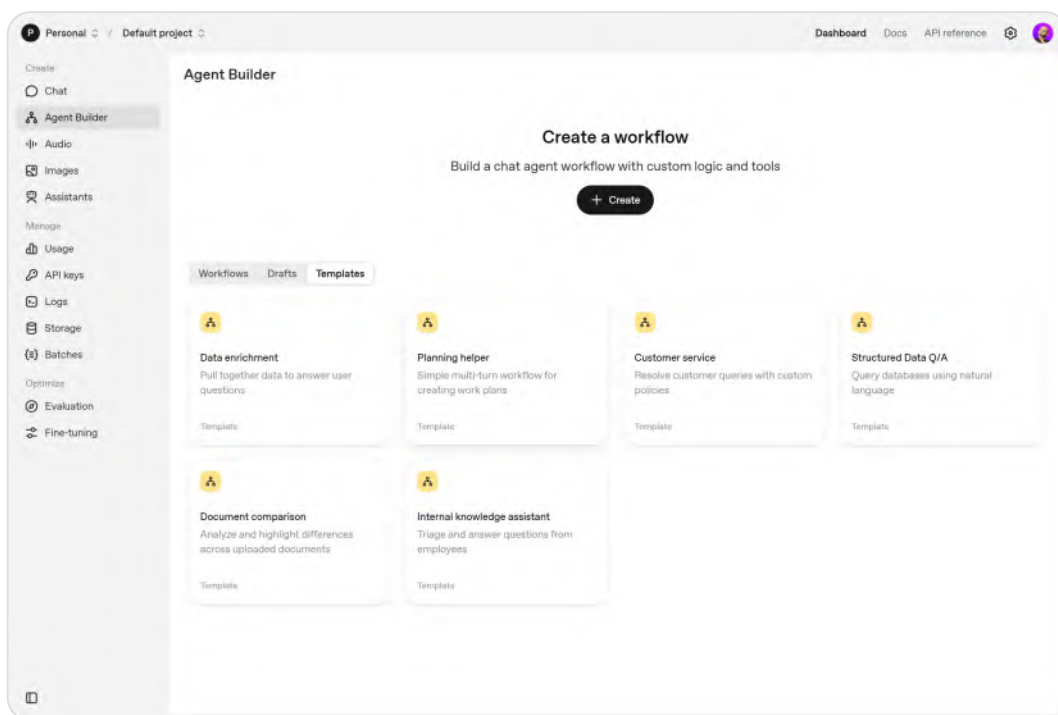
ด้วยเครื่องมือเหล่านี้ นักพัฒนาสามารถ **ประเมิน ปรับแต่ง และ
พัฒนาเอเจนต์ได้อย่างต่อเนื่อง** เพื่อให้มั่นใจว่าเอเจนต์ของคุณพร้อม
ใช้งานในระดับ production และตอบสนองความต้องการของผู้ใช้ได้
อย่างมีประสิทธิภาพสูงสุด

ในบทถัดไป เราจะไปทำความรู้จักกับ **Agent Builder** เครื่องมือ
สำคัญในชุด AgentKit ที่ช่วยให้คุณสามารถ “สร้าง – ทดสอบ – ส่ง
ออก” เวิร์กโฟลว์เอเจนต์หลายขั้นตอนได้อย่างง่ายดาย

บทที่ 2: แนะนำ Agent Builder

Agent Builder คือเครื่องมือหลักในชุด **AgentKit** ที่ช่วยให้คุณ
สามารถ “สร้าง–ทดสอบ–ส่งออก” เวิร์กโฟลว์ของเอเจนต์หลายชั้น
ตอนได้อย่างง่ายดายผ่านอินเทอร์เฟซแบบแผนภาพ คุณสามารถเข้า
ใช้งาน Agent Builder ได้ที่

<https://platform.openai.com/agent-builder>



หน้าเว็บ Agent Builder

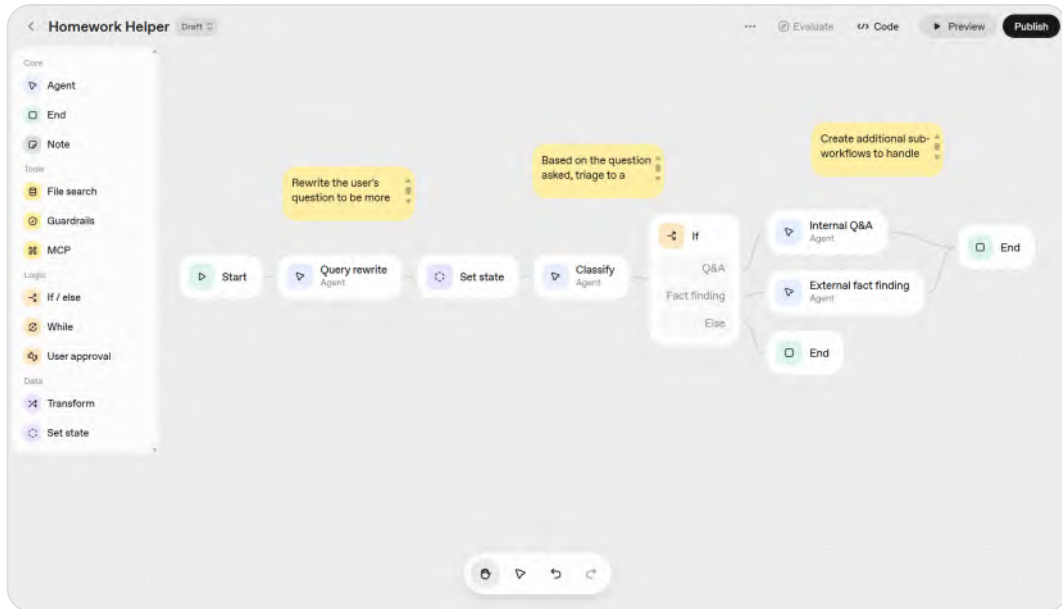
Agent Builder มอบเครื่องมือออกแบบเวิร์กโฟลว์เอเจนต์ที่ใช้งานง่าย
ผ่านพื้นที่ทำงานแบบ canvas

คุณสามารถเริ่มต้นสร้างสรรค์ได้ทันทีจาก **เทมเพลตสำเร็จรูป** หรือ **ลากและวาง โหนด** เพื่อกำหนดขั้นตอนการทำงานที่ซับซ้อนของเอเจนต์อย่างอิสระ พร้อมทั้งระบุประเภทข้อมูลเข้า–ออก (typed inputs and outputs) เพื่อความแม่นยำ และสามารถ **ทดสอบการทำงานแบบเรียลไทม์** ได้ในตัว

เมื่อเวิร์กโฟลว์ของคุณพร้อมสำหรับการใช้งานจริง คุณมีทางเลือกในการนำไปใช้ 2 รูปแบบหลัก คือ **ฝังเข้ากับแอปพลิเคชันของคุณโดยใช้ ChatKit** หรือ **ดาวน์โหลดโค้ด SDK** เพื่อนำไปรันและใช้งานบนระบบของคุณเอง (Self-Hosting)

แนะนำ Agents และ Workflows

ก่อนที่คุณจะสร้างเอเจนต์คุณต้องเริ่มจากการสร้าง “เวิร์กโฟลว์” (workflow) ก่อน **เวิร์กโฟลว์** คือการรวมกันของเอเจนต์ เครื่องมือ และตรรกะควบคุม (control-flow logic) ห่อหุ้มขั้นตอนของการทำงาน ตั้งแต่การรับอินพุต การประมวลผล ไปจนถึงการตอบกลับผลลัพธ์ โดยทั้งหมดอยู่ในรูปแบบของโค้ดที่สามารถนำไปใช้งานได้จริง



ตัวอย่างเวิร์กโฟลว์

กระบวนการสร้างเอเจนต์เพื่อจัดการงานหนึ่งๆ ประกอบด้วย 3 ขั้นตอนหลัก

1. **ออกแบบเวิร์กโฟลว์ใน Agent Builder** — กำหนดโครงสร้างและลำดับการทำงานของเอเจนต์
2. **เผยแพร่ (Publish) เวิร์กโฟลว์ของคุณ** — เวิร์กโฟลว์จะถูกบันทึกเป็นอ็อบเจกต์ที่มี ID และระบบเวอร์ชัน
3. **นำเวิร์กโฟลว์ไปใช้ (Deploy)** — นำ Workflow ID ไปใช้ใน ChatKit หรือดาวน์โหลดโค้ดเพื่อรันด้วยตัวเอง

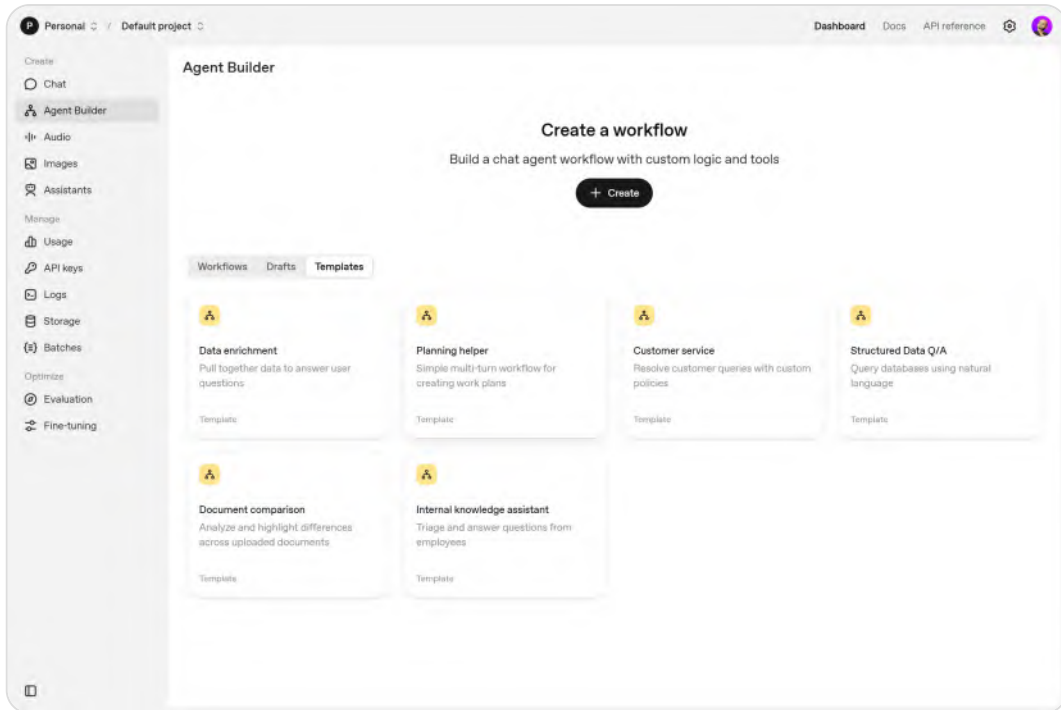
การประกอบ Workflow

การสร้างเวิร์กโฟลว์โดยใช้ Agent Builder เป็นการเพิ่ม **โหนด (nodes)** และเชื่อมต่อโหนด ซึ่งแต่ละโหนดคือ **ขั้นตอนหนึ่งในกระบวนการ** โดย **เส้นเชื่อมระหว่างโหนด (edge)** จะแสดงการส่งต่อข้อมูล เป็น **เส้นระบุชนิดข้อมูล (typed edge)** เพื่อให้มั่นใจได้ว่าข้อมูลที่ส่งต่อมีรูปแบบถูกต้อง

คุณสามารถคลิกที่โหนดเพื่อดูหรือแก้ไขข้อมูลเข้า-ออก (inputs/outputs) ตรวจสอบสัญญาข้อมูล (data contract) ระหว่างขั้นตอน และยืนยันว่าโหนดถัดไปได้รับข้อมูลที่ต้องการครบถ้วน

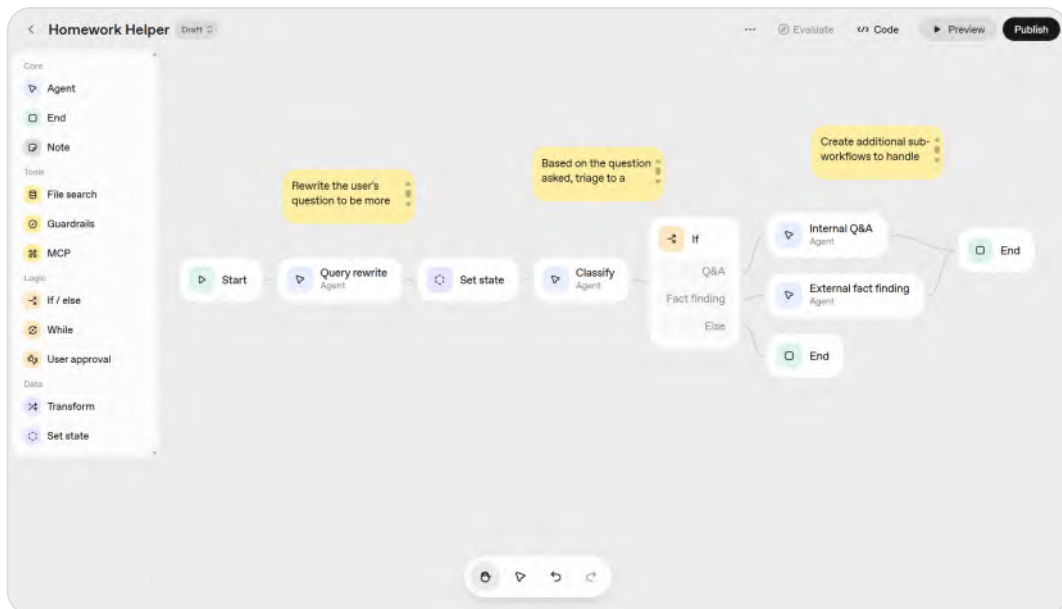
เวิร์กโฟลว์และเทมเพลต

Agent Builder มีเทมเพลตให้เลือกใช้หลายตัว ซึ่งเป็นเวิร์กโฟลว์ยอดนิยม คุณสามารถเริ่มต้นสร้างเวิร์กโฟลว์จากเทมเพลตเพื่อศึกษาวิธีเชื่อมต่อโหนดต่างๆ หรือจะสร้างจากศูนย์ก็ได้



เทมเพลตเวิร์กโฟลว์

ตัวอย่างเช่น เวิร์กโฟลว์ **“ผู้ช่วยทำการบ้าน” (Homework Helper)** ใช้เอเจนต์ในการรับคำถามจากผู้ใช้ ปรับแต่งคำถามให้เหมาะสม ส่งต่อไปยังเอเจนต์เฉพาะทางเพื่อหาคำตอบ และส่งผลลัพธ์สุดท้ายกลับไปยังผู้ใช้



เวิร์กโฟลว์ผู้ช่วยทำการบ้าน

ประเภทโหนด

โหนดแต่ละตัวเปรียบเสมือน “ชิ้นส่วนเลโก้” ที่มีหน้าที่เฉพาะ เมื่อประกอบกันแล้วจะกลายเป็นเวิร์กโฟลว์ที่สมบูรณ์ โดยโหนดใน Agent Builder แบ่งออกเป็น 4 ประเภทหลัก ได้แก่

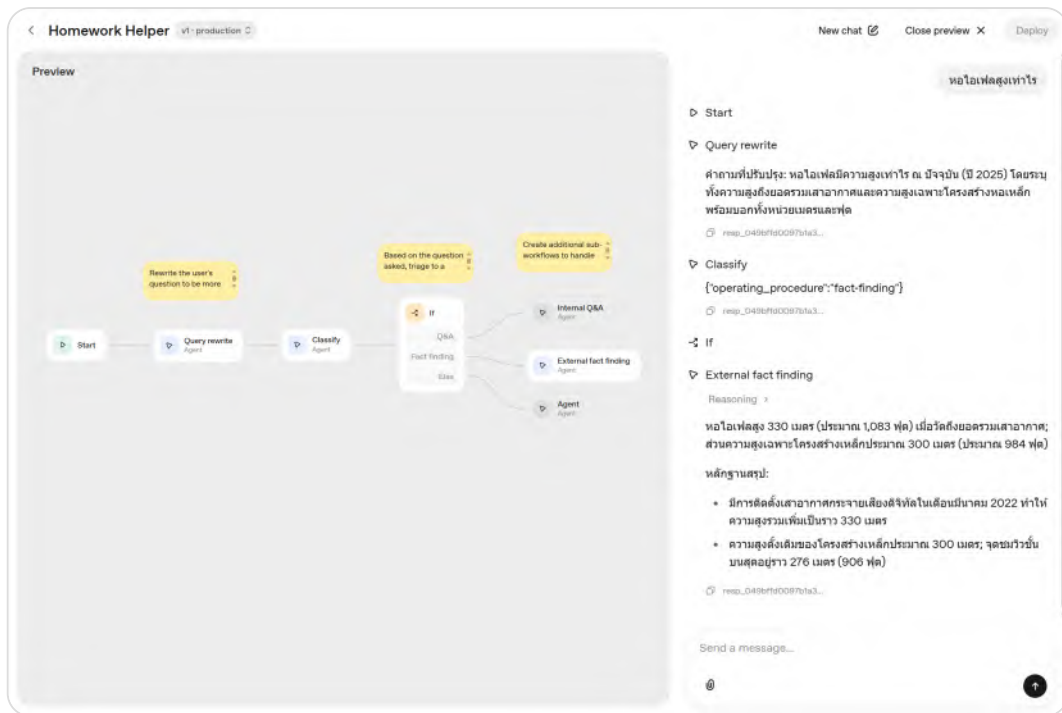
- **โหนดหลัก (Core Nodes):** ส่วนพื้นฐานของทุกเวิร์กโฟลว์ เช่น Start node และ Agent node ซึ่งเป็นจุดเริ่มต้นและหัวใจของกระบวนการทั้งหมด
- **โหนดเครื่องมือ (Tool Nodes):** ใช้สำหรับเชื่อมต่อกับโลกภายนอก เช่น ดึงข้อมูล ทำงานกับไฟล์ หรือเชื่อมต่อกับระบบอื่น เช่น File Search, Guardrails และ MCP (Marketplace Connectors Platform)

- **โหนดตรรกะ (Logic Nodes):** ช่วยเพิ่มความยืดหยุ่นด้วยการสร้างเงื่อนไขและการตัดสินใจ เช่น If/Else, While, Human Approval เป็นต้น
- **โหนดข้อมูล (Data Nodes):** ใช้จัดการหรือแปลงข้อมูลที่ไหลในเวิร์กโฟลว์ เช่น Transform หรือ Set State

รายละเอียดของโหนดแต่ละประเภทจะอธิบายอย่างละเอียดในบทถัดไป

การทดสอบเวิร์กโฟลว์และดีบั๊ก

ในระหว่างการสร้าง คุณสามารถทดสอบเวิร์กโฟลว์ได้ทันทีด้วยฟีเจอร์ **Preview** เพื่อดูการทำงานจริงของแต่ละโหนด สามารถแนบไฟล์ตัวอย่าง ทดสอบการไหลของข้อมูล และสังเกตพฤติกรรมของเวิร์กโฟลว์แบบโต้ตอบได้



ดูตัวอย่างและดีบั๊ก

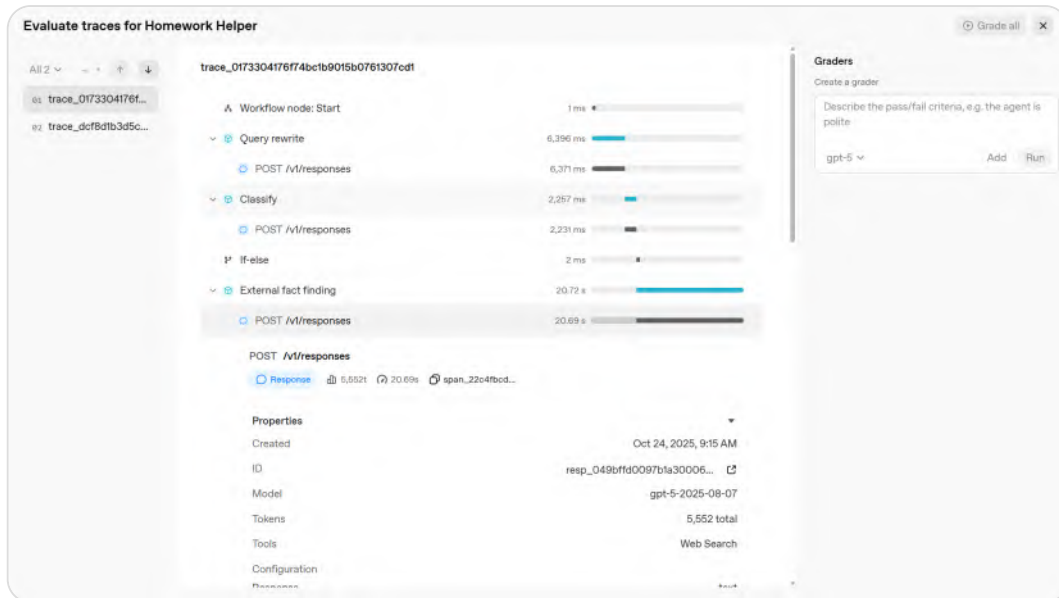
ความปลอดภัยและความเสี่ยง

การสร้างเวิร์กโฟลว์ของเอเจนต์มักเกี่ยวข้องกับข้อมูลสำคัญ ซึ่งอาจเสี่ยงต่อการโจมตีแบบ **prompt injection** หรือการรั่วไหลของ **ข้อมูล** เราจะมาดูแนวทางด้านความปลอดภัยอย่างละเอียด ในบทที่ 4 เพื่อช่วยคุณลดความเสี่ยงเหล่านี้และออกแบบเวิร์กโฟลว์ที่เชื่อถือได้

การประเมินคุณภาพ

Agent Builder มีเครื่องมือช่วยประเมินคุณภาพเวิร์กโฟลว์ในตัว โดยคุณสามารถคลิก **Evaluate** ที่แถบนำทางด้านบน เพื่อรัน **trace**

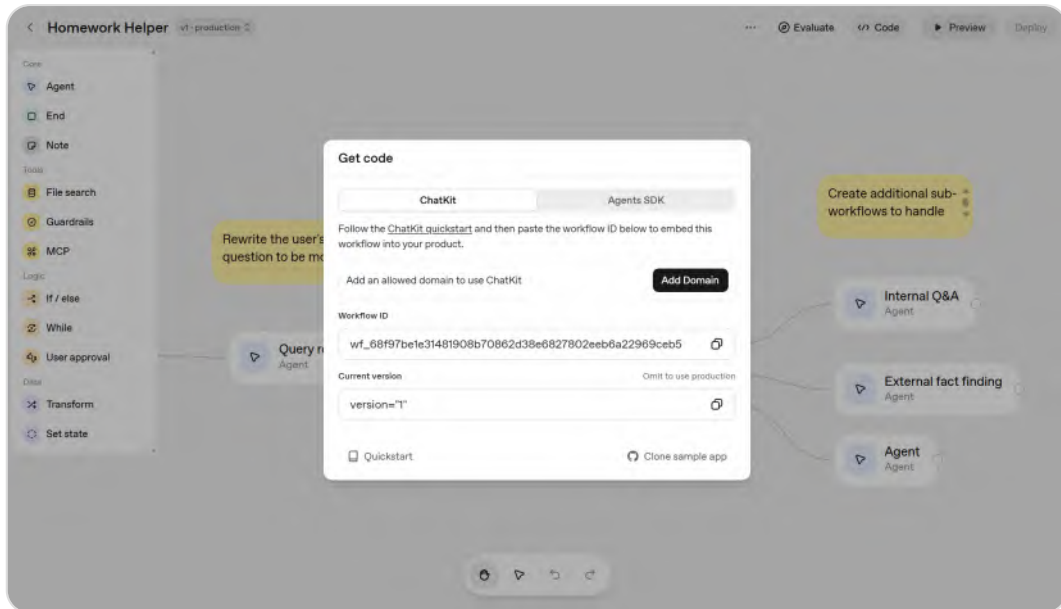
graders บนชุดข้อมูล trace ที่เลือก และดูผลการประเมินประสิทธิภาพของเวิร์กโฟลว์โดยรวม



การประเมินเวิร์กโฟลว์

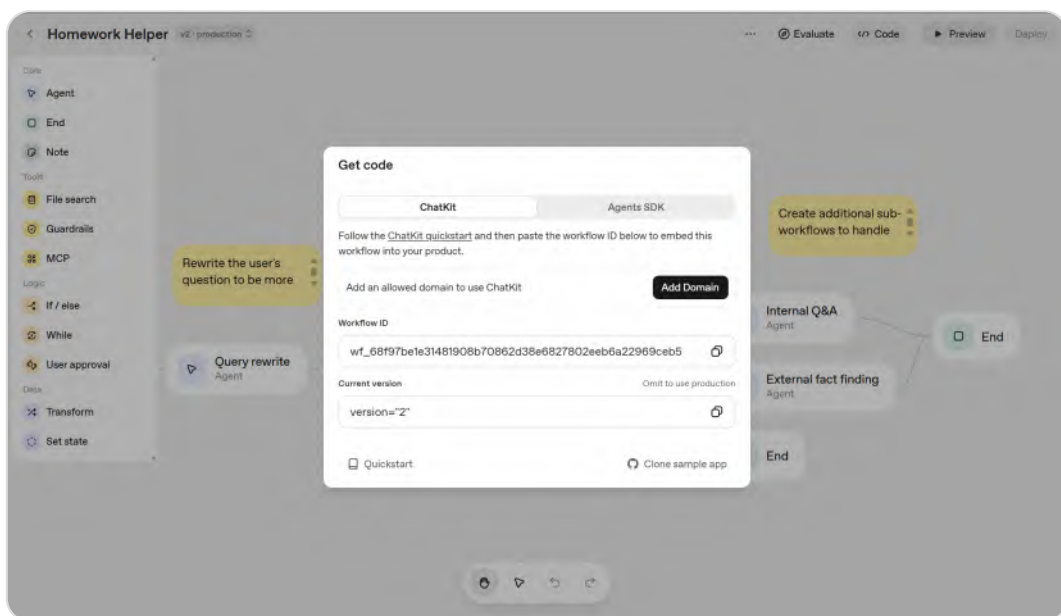
การเผยแพร่เวิร์กโฟลว์

Agent Builder จะบันทึกงานของคุณโดยอัตโนมัติระหว่างการทำงานเมื่อพร้อมแล้ว ให้ **Publish** เวิร์กโฟลว์เพื่อสร้างเวอร์ชันหลักใหม่ (snapshot) ซึ่งสามารถนำไปใช้งานใน **ChatKit** ได้ทันที ซึ่ง ChatKit เป็น UI เฟรมเวิร์กใช้สำหรับฝังกล่องแชทลงในแอปพลิเคชันของคุณ



เวิร์กโฟลว์เวอร์ชัน 1

นอกจากนี้ คุณยังสามารถสร้างเวอร์ชันใหม่ หรือเรียกใช้เวอร์ชันเก่า ผ่าน API ได้ตามต้องการ

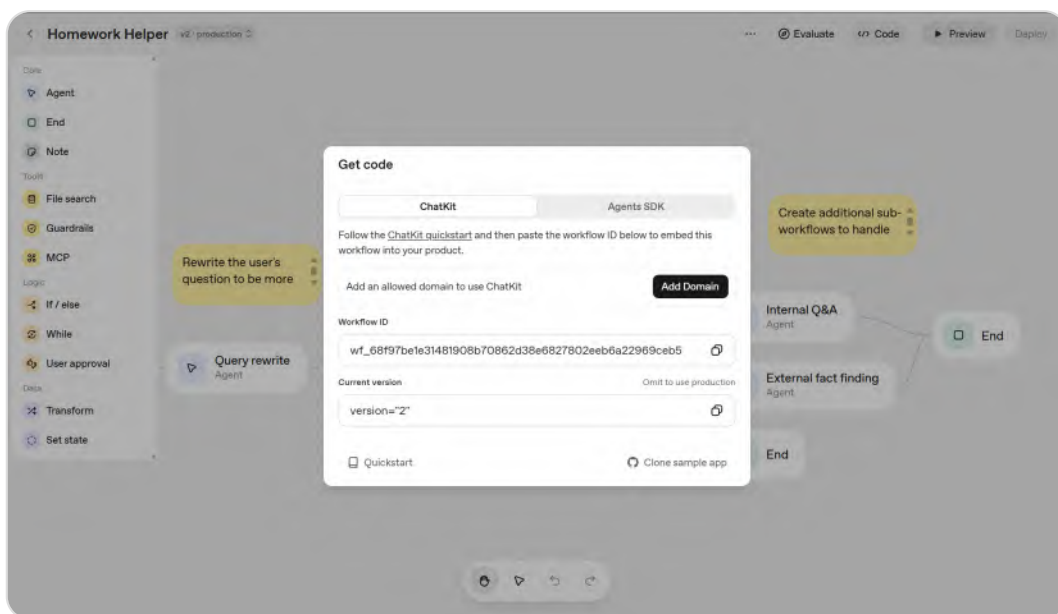


เวิร์กโฟลว์เวอร์ชัน 2

การใช้งานร่วมกับแอปพลิเคชัน

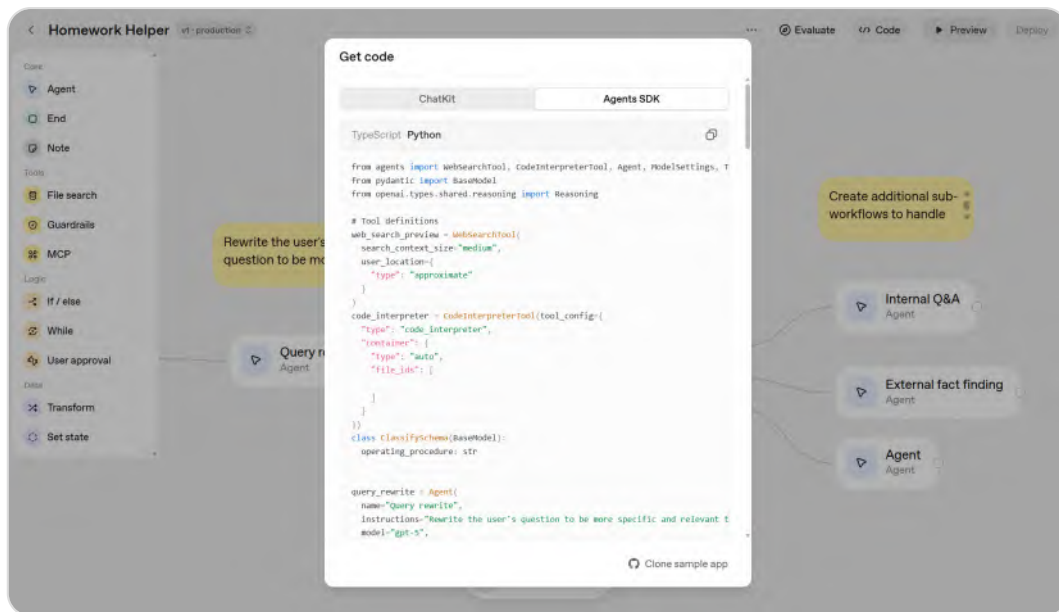
เมื่อเวิร์กโฟลว์พร้อมสำหรับ production ให้คลิกแท็บ **Code** จากนั้นคุณมีสองทางเลือกในการ deploy

- **ChatKit:** วิธีที่ง่ายที่สุด เพียงใส่ Workflow ID ลงใน ChatKit และฝังแชทเอเจนต์ลงในแอปพลิเคชัน



เรียกใช้ Workflow ID โดยใช้ ChatKit

- **Advanced Integration:** สำหรับผู้ที่ต้องการควบคุมมากขึ้น สามารถคัดลอกโค้ดของเวิร์กโฟลว์ไปใช้กับโครงสร้างพื้นฐานของคุณเอง ผ่าน **Agents SDK** เพื่อสร้างและปรับแต่งประสบการณ์การใช้งานได้อย่างอิสระ



โค้ดจาก Agent Builder

ตอนนี้คุณได้เรียนรู้ขั้นตอนการสร้างเวิร์กโฟลว์ของเอเจนต์แล้ว ในบทถัดไป เราจะเจาะลึกถึง **รายละเอียดของโหนดแต่ละประเภท** ที่คุณสามารถใช้ภายใน Agent Builder เพื่อสร้างเอเจนต์ที่ทรงพลังและยืดหยุ่นยิ่งขึ้น

บทที่ 3: ทำความรู้จักกับ Nodes ใน Agent Builder

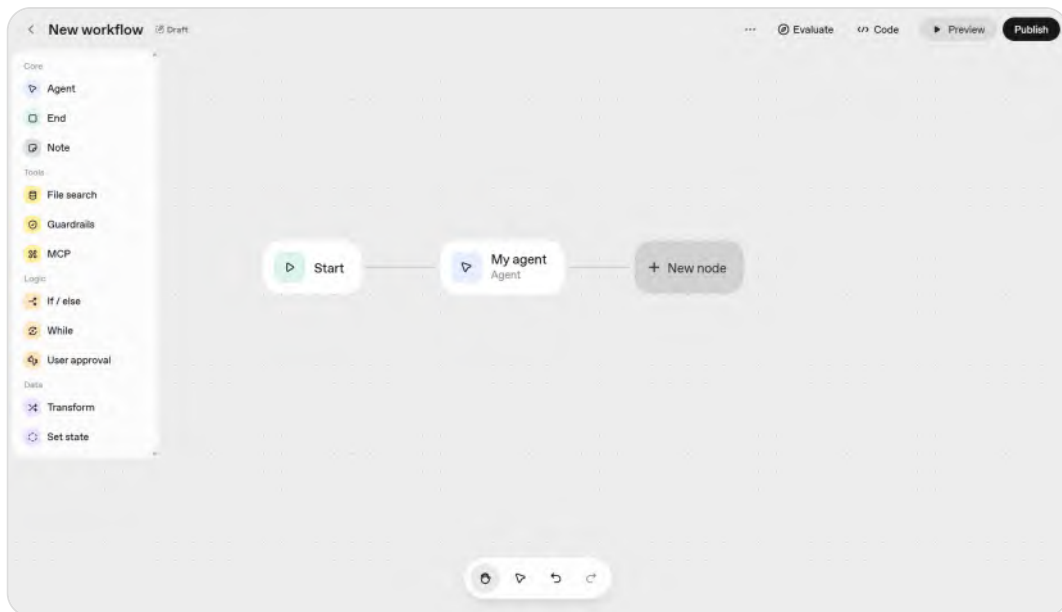
Agent Builder คือเครื่องมือสำคัญในการสร้างเวิร์กโฟลว์ของเอเจนต์ (Agentic Workflows) ซึ่งช่วยให้องค์ประกอบต่างๆ ของเอเจนต์สามารถทำงานเป็นลำดับขั้นตอนและสื่อสารกันได้อย่างมีประสิทธิภาพ

หัวใจของระบบนี้อยู่ที่สิ่งที่เรียกว่า **“โนด” (Nodes)** ซึ่งเป็นหน่วยพื้นฐานที่มีหน้าที่เฉพาะด้าน แต่ละโนดเปรียบเสมือน **ชิ้นส่วนเลโก้** ที่สามารถนำมาต่อเข้าด้วยกันได้อย่างอิสระ และเมื่อประกอบเข้ากันอย่างเหมาะสม ก็จะสร้างกระบวนการทำงานที่มีความซับซ้อน ยืดหยุ่น และทรงพลังขึ้นมาได้

ในบทนี้ เราจะมาทำความเข้าใจว่าโนดใน Agent Builder มีประเภทใดบ้าง แต่ละแบบมีหน้าที่อย่างไร และควรเลือกใช้ “ชิ้นส่วน” ใดให้เหมาะกับสถานการณ์ เพื่อออกแบบเวิร์กโฟลว์ของเอเจนต์ให้มีประสิทธิภาพสูงสุด

โนดหลัก (Core Nodes)

โนดหลัก (Core Nodes) คือกลุ่มโนดพื้นฐาน และเป็นหัวใจของทุกเวิร์กโฟลว์ ทุกระบบจะต้องมีอย่างน้อย 2 โหนดคือ **Start** และ **Agent**



ตัวอย่างโหนดหลัก

โหนด Start

โหนด Start คือ **ประตูทางเข้า** ของเวิร์กโฟลว์ ทำหน้าที่รับข้อมูลเริ่มต้นเข้ามาในระบบ โดยเฉพาะในเวิร์กโฟลว์ประเภทแชท (Chat Workflow) โหนด Start จะทำสิ่งเหล่านี้โดยอัตโนมัติ

- บันทึกข้อความล่าสุดจากผู้ใช้ไว้ในประวัติการสนทนา
- สร้างตัวแปร `input_as_text` เพื่อให้โหนดอื่นสามารถนำข้อความนั้นไปใช้ต่อ