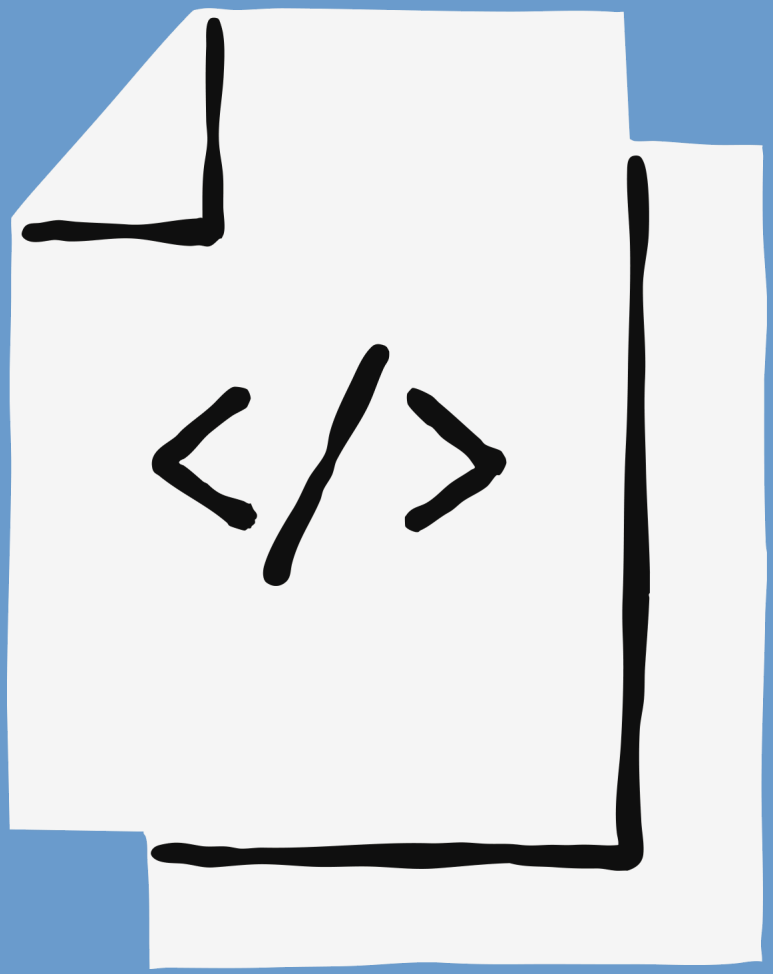
Xây dựng AI Agent hiệu quả: Mẫu kiến trúc và khung triển khai

Tăng tốc chuyển đổi AI cho doanh nghiệp của bạn với những chiến lược đã được kiểm chứng từ khách hàng và đội ngũ nội bộ của Anthropic.

Claude

Phiên bản tiếng Việt • Biên dịch & Việt hóa bởi Phong Hồ

Mục lục



Chương 1

Tóm tắt tổng quan

Chương 1

Tóm tắt tổng quan

AI tạo sinh trả lời câu hỏi. AI agent giải quyết vấn đề.

Với các công ty ở mọi ngành, agent mở ra khả năng mở rộng vận hành mà các công cụ tự động hóa hiện tại không bao giờ làm được: giải quyết vấn đề mở, ra quyết định linh hoạt, và xử lý các quy trình nhiều bước phức tạp khi con đường phía trước chưa được định sẵn.

Nhiều tổ chức đã thu được kết quả đáng kể từ agent tự hành trong môi trường production. Chẳng hạn **Coinbase**, sàn giao dịch tiền mã hóa hàng đầu xử lý 226 tỷ USD khối lượng giao dịch mỗi quý, đã xây dựng hệ thống chăm sóc khách hàng agentic chạy trên Claude. Các agent của họ xử lý hàng nghìn tin nhắn mỗi giờ với độ sẵn sàng 99,99% - yếu tố sống còn khi khách hàng cần truy cập tiền của mình mọi lúc. Nền tảng này đã sản sinh 35-50 ứng dụng AI nội bộ, thay đổi cách công ty phục vụ hàng triệu người dùng toàn cầu.

Tines, nền tảng điều phối và tự động hóa quy trình cho đội ngũ bảo mật và IT, xây dựng hệ thống quy trình agentic với Claude. Agent của họ xử lý logic quy trình một cách linh hoạt ngay trong lúc chạy, thu gọn các chuỗi thao tác bảo mật nhiều bước phức tạp thành thao tác một agent duy nhất, giúp thời gian tạo ra giá trị nhanh hơn 100 lần.

Còn **Gradient Labs**, công ty xây dựng agent vận hành khách hàng cho ngành dịch vụ tài chính, đã triển khai agent hỗ trợ khách hàng bằng Claude có khả năng hiểu câu hỏi của khách trong đúng ngữ cảnh và thực thi các quy trình vận hành chuẩn. Đạt tỷ lệ giải quyết 80-90%, agent của họ gánh được khối lượng việc phức tạp mà gần như không cần con người can thiệp, để nhân viên tập trung vào xây dựng quan hệ khách hàng và các việc chiến lược khác.

AI agent mở ra vô vàn khả năng cho tổ chức thuộc mọi quy mô và lĩnh vực, nhưng triển khai chúng đòi hỏi cân nhắc kỹ về mẫu kiến trúc, quản lý chi phí và quản trị vận hành.

Bài toán kinh doanh của AI agent

Hãy hình dung AI agent như một trợ lý số thông minh có thể làm việc độc lập để giải quyết các bài toán kinh doanh phức tạp, bằng cách dùng các **tool** kết nối vào hệ thống thật của bạn. Về bản chất, **AI agent** là bước tiến hóa cao hơn của mô hình ngôn ngữ lớn: nó tự định hướng quá trình xử lý và **cách dùng tool** của chính mình để hoàn thành những nhiệm vụ phức tạp.

Tự động hóa truyền thống cần kịch bản viết sẵn cứng nhắc với từng bước được vạch trước. Agent làm việc theo cách khác. Chúng đánh giá nhiệm vụ, chọn tool phù hợp, thử các hướng tiếp cận, đánh giá kết quả và điều chỉnh chiến lược khi cần - rất giống cách một nhân viên giỏi xử lý dự án lạ. Ví dụ, một agent tiếp nhận các ca hỗ trợ khách hàng leo thang có thể đọc vấn đề, kiểm tra lịch sử tài khoản, tra cứu kho tri thức, soạn câu trả lời cá nhân hóa và gọi thêm chuyên gia khi cần, tất cả không cần con người can thiệp.

Điều làm các hệ thống này mạnh là khả năng tự suy luận và **tự chọn tool**, cộng với khả năng phục hồi sau lỗi và bám đuổi mục tiêu đến cùng. Khác với quy trình truyền thống nơi các đường code định sẵn điều phối tương tác AI, agent giữ quyền kiểm soát linh hoạt đối với quá trình ra quyết định của mình, thích ứng theo phản hồi từ môi trường và kết quả trung gian.

Chính vì vậy, chúng đặc biệt giá trị khi cần mở rộng các nghiệp vụ phức tạp mà không thể định sẵn từng bước, như ứng phó sự cố, phân tích dữ liệu, quy trình onboarding khách hàng, hay quy trình phát triển phần mềm nơi kiểm thử tự động tạo thành vòng phản hồi để giải quyết vấn đề theo từng vòng lặp.

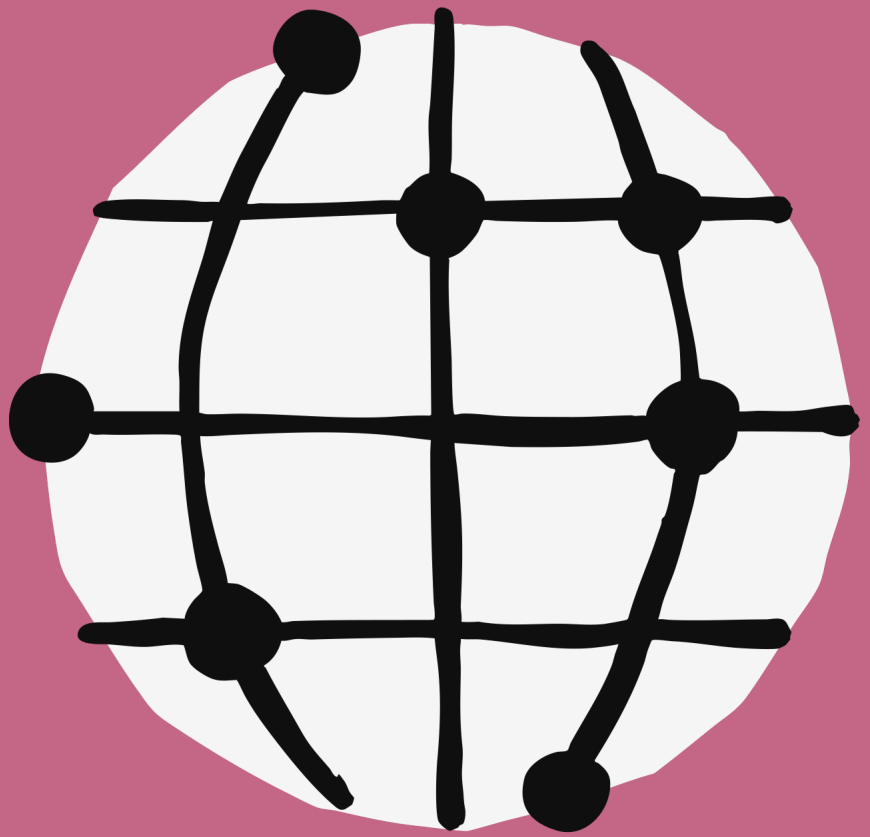
Các tổ chức đang đạt được gì

Những tổ chức triển khai agent đang thu về các con số rất đáng chú ý.

Tại một ngân hàng bán lẻ, chẳng hạn, AI agent đã thay đổi hẳn cách tạo bản ghi nhớ rủi ro tín dụng. Việc trước đây các chuyên viên quan hệ khách hàng phải mất hàng tuần rà soát thủ công mười nguồn dữ liệu khác nhau, giờ mang lại **mức tăng năng suất 20 đến 60 phần trăm** và rút ngắn 30 phần trăm thời gian xử lý hồ sơ tín dụng. Một nhà sản xuất thiết bị châu Âu với doanh thu hơn 10 tỷ euro đã vạch chiến lược AI agentic của họ và nhận thấy

- từ các đội ngũ đang vận hành những hệ thống này trong production
- Các mẫu kiến trúc từ agent đơn đến điều phối đa agent, kèm chỉ dẫn rõ ràng để ghép đúng bài toán của bạn với (các) mẫu phù hợp
- Yêu cầu kỹ thuật gồm năng lực API, tích hợp tool và quản lý bộ nhớ cho agent sẵn sàng production thực sự chạy được ở quy mô lớn
- Khung bảo mật và tuân thủ để bảo vệ dữ liệu nhạy cảm trong khi quản lý những rủi ro đặc thù của hệ thống tự hành
- Chiến lược triển khai để xây đội ngũ và hạ tầng lớn lên cùng các cải tiến của mô hình (thay vì chống lại chúng)
- Các chỉ báo sẵn sàng cho tương lai giúp bạn xây hệ thống ngày càng mạnh khi mô hình nền tảng tốt lên - mà không ngày càng phức tạp

Giờ đi vào chi tiết.



Chương 2

Các tình huống sử dụng và ứng dụng phổ biến của AI agent

Chương 2

Các tình huống sử dụng và ứng dụng phổ biến của AI agent

Để hiểu agent tạo ra giá trị kinh doanh thực sự ở đâu, hãy xem các tổ chức đang triển khai chúng thế nào. Các ca triển khai thực tế hé lộ những khuôn mẫu có thể định hướng quyết định chiến lược của chính bạn và giúp bạn nhận ra các cơ hội tác động lớn nhất trong tổ chức mình.

Lập trình

Tăng tốc phát triển trên hệ thống doanh nghiệp: **Augment Code** dùng Claude trên Vertex AI của Google Cloud để giúp lập trình viên điều hướng những codebase phức tạp với hàng triệu dòng code phụ thuộc lẫn nhau. Một khách hàng doanh nghiệp đã hoàn thành trong 2 tuần dự án mà CTO của họ ước tính phải mất 4-8 tháng, còn thời gian onboarding lập trình viên mới rút từ vài tuần xuống 1-2 ngày. Nền tảng này giúp các đội hiểu hệ thống phần mềm phức tạp nhanh hơn, nhờ đó viết, tài liệu hóa và bảo trì code hiệu quả hơn.

Phân tích dữ liệu

Khai phá dữ liệu observability bằng hội thoại: Grafana dùng Claude xây trợ lý thông minh giúp mọi cấp độ kỹ năng trong đội, từ CTO đến kỹ sư mới, **khai thác dữ liệu observability bằng ngôn ngữ tự nhiên**. Người dùng có thể hỏi những câu như "Độ trễ request của dịch vụ checkout của tôi là bao nhiêu?" và Claude tự tìm các chỉ số liên quan rồi dựng truy vấn PromQL và LogQL phù hợp.

Hỗ trợ khách hàng và vận hành

Hỗ trợ tự động tỷ lệ giải quyết cao ở quy mô lớn: Fin, AI agent của Intercom chạy trên Claude, đạt **tỷ lệ giải quyết tới 86%** với chất lượng câu trả lời ngang con người trên hơn 25.000 khách hàng. Nền tảng đạt tỷ lệ tự giải quyết trung bình 51% ngay khi chưa tùy biến, rút thời gian phản hồi từ 30 phút xuống vài giây, và hỗ trợ hơn 45 ngôn ngữ.

Phối hợp hỗ trợ giữa AI và con người: Assembled dùng Claude cho nền tảng Assist của họ, đạt **mức tăng 20% độ hài lòng khách hàng** trong khi giảm chi phí hỗ trợ, giảm hơn 50% số ca leo thang và tăng hơn 30% số ca giải quyết mỗi giờ. Cách tiếp cận của họ tập trung xử lý các vấn đề Tier 2+ phức tạp thay vì chỉ chặn bớt các câu hỏi đơn giản.

Pháp lý

Tri thức pháp lý doanh nghiệp ở quy mô lớn: Thomson Reuters dùng Claude trong Amazon Bedrock để vận hành CoCounsel, mang **chuyên môn của hơn 3.000 chuyên gia** và hơn 150 năm nội dung chuẩn mực đến giới luật và thuế. Nền tảng xử lý các hợp đồng và hồ sơ thuế phức tạp với độ chính xác nghiêm ngặt qua kiểm định chuyên gia; khách hàng cho biết họ "dễ dàng thấy nó cắt giảm một nửa thời gian, thậm chí hơn" và mô tả hiệu suất của CoCounsel là "kinh ngạc" - giải phóng các chuyên gia để "tập trung vào công việc tầm cao hơn, chiến lược hơn."

AI pháp lý linh hoạt với khả năng bám sát chỉ dẫn vượt trội: Legora dùng Claude cho toàn bộ nền tảng pháp lý của mình, **đạt hiệu năng cao hơn 18%** trên bộ đánh giá pháp lý quy mô lớn của riêng họ qua các tác vụ phức tạp. Mức tiến bộ của Claude Sonnet đến từ "sự ổn định trên các tác vụ và tài liệu lớn cùng khả năng bám chính xác những chỉ dẫn phức tạp", giúp Legora xây các quy trình agentic linh hoạt thích ứng theo từng mảng hành nghề và yêu cầu khách hàng, hỗ trợ luật sư "rà soát và nghiên cứu chuẩn xác, soạn thảo thông minh hơn, cộng tác liền mạch."

Marketing

Quảng cáo đa nền tảng tự động ở quy mô lớn: Advolve dùng Claude điều phối toàn bộ quy trình thu hút khách hàng số của họ, quản lý hàng triệu quảng cáo cùng lúc trên nhiều nền tảng với kiểm tra dữ liệu thời gian thực và phân bổ ngân sách linh hoạt. Hệ thống giảm 90% thời gian thao tác vận hành và **tăng 15% tỷ suất lợi nhuận trên chi phí quảng cáo** (ROAS) của khách hàng; nền tảng của họ đạt "ROAS ngang chuyên gia khi quản lý ngân sách hàng triệu đô trong chưa đầy 30 ngày" cho các khách hàng doanh nghiệp lớn với ngân sách quảng cáo vượt 100 triệu USD.

Tiêu điểm ngành: Dịch vụ tài chính

Phát hiện gian lận và đánh giá rủi ro tự động: Inscribe dùng Claude vận hành các AI Risk Agent giúp giảm 20 lần thời gian rà soát gian lận - từ 30 phút xuống 90 giây - **trong khi tăng sản lượng 70 lần** trong các ví dụ khách hàng. AI Fraud Analyst phát hiện gian lận trong ảnh và PDF, xác minh thông tin người nộp hồ sơ qua kiểm tra KYC và KYB, phát hiện giao dịch rủi ro và xuất báo cáo rủi ro kiểm toán được trong khoảng 90 giây. Điều này giúp các định chế tài chính mở rộng tiếp cận tín dụng đến những nhóm xứng đáng nhưng chưa được phục vụ, gồm người ít lịch sử tín dụng, chưa có tài khoản ngân hàng hay "vô hình" với hệ thống tín dụng.

Chương 3

Các mẫu kiến trúc phổ biến

Chương 3

Các mẫu kiến trúc phổ biến

Các tình huống trên cho thấy tiềm năng to lớn của agent ở nhiều ngành, nhưng triển khai thành công còn phụ thuộc hoàn toàn vào việc chọn đúng mô hình, công nghệ và cách tiếp cận kiến trúc. Một agent hỗ trợ khách hàng xử lý câu hỏi thường nhật cần thiết kế khác hẳn một hệ thống nghiên cứu đa lĩnh vực phân tích các bộ dữ liệu phức tạp.

Hiểu các mẫu kiến trúc này giúp bạn cân đối độ phức tạp kỹ thuật với yêu cầu kinh doanh, đồng thời tránh kiểu thiết kế thừa (over-engineering) làm đội chi phí mà không mang lại giá trị tương xứng.

Hãy bắt đầu với những nguyên tắc thiết kế nền tảng dẫn lối cho các ca triển khai AI agent thành công.

Cách làm tốt nhất khi thiết kế agent

Bắt đầu đơn giản, mở rộng thông minh. Như đã bàn trong **Building Effective AI Agents**, chúng tôi khuyên các đội bắt đầu với agent đơn nhiệm làm tốt một việc, rồi dần phát triển thành hệ thống tinh vi hơn khi yêu cầu của bạn tiến hóa. Hệ thống đơn giản rẻ hơn khi vận hành (ít token, ít tài nguyên tính toán), dễ gỡ lỗi hơn khi có sự cố, và cho bạn các chỉ số rõ ràng thực sự gắn với kết quả kinh doanh.

Chọn đúng mô hình cho đúng việc. Có rất nhiều mô hình AI với năng lực rất khác nhau, và chọn đúng là điều then chốt.

Chìa khóa là cân bằng ba yếu tố: năng lực, tốc độ và chi phí. Hãy nghĩ như chọn dụng cụ trong hộp đồ nghề: bạn sẽ không dùng búa tạ để treo khung ảnh, và cũng không dùng búa đóng đinh mũ để phá tường. Chẳng hạn, nếu đang xây framework lập trình đa agent hay phân tích tài chính phức tạp, bạn sẽ muốn **mô hình mạnh nhất hiện có**. Nhưng nếu bạn đang xử lý hàng nghìn ticket hỗ trợ đơn giản hay trích dữ liệu từ biểu mẫu, một mô hình nhẹ và nhanh hơn vẫn làm tốt y vậy với chi phí chỉ bằng một phần nhỏ.

Phổ năng lực trải từ các mô hình tối ưu cho tác vụ suy luận phức tạp nhất xuống các mô hình thiết kế cho khối lượng lớn, đơn giản. Nghĩa là bạn có thể ghép đúng tình huống sử dụng với đúng mức năng lực. Chạy một việc đơn giản qua mô hình cao cấp không chỉ lãng phí - nó còn chậm hơn và đắt hơn khi nhân rộng. Khi bạn xử lý hàng trăm, hàng nghìn request, những khác biệt đó cộng dồn rất nhanh.

Thiết kế module hóa. Lĩnh vực này chuyển động nhanh; năng lực và tính năng mới xuất hiện liên tục. Hãy thiết kế hệ thống theo hướng module để có thể nâng cấp năng lực agent mà không phải đập đi xây lại hạ tầng. Kiến trúc của bạn nên uốn theo tiến bộ, chứ không gãy vì nó.

- Tính module của agent, chẳng hạn, có thể dựa trên mẫu lắp ghép (composition) trong đó:
- Prompt được định nghĩa trong file cấu hình hoặc thư viện tập trung
- Tool là các module rời có thể tái sử dụng

Agent được định nghĩa khi cần, chỉ dùng đúng các tool và/hoặc tài nguyên cần thiết để hoàn thành nhiệm vụ được giao.

Mẫu lắp ghép này có thể gồm các module riêng cho tìm kiếm web, truy vấn cơ sở dữ liệu và soạn email, cùng các mẫu prompt cho những phong cách suy luận khác nhau (phân tích, sáng tạo, kỹ thuật) hoặc cho từng vai trò. Theo cách đó, bạn nhanh chóng định nghĩa các agent cần thiết trong lúc phát triển và chọn tài nguyên định sẵn khi cần.

Những agent module hóa như vậy có thể lớn lên một cách tự nhiên, nhất là khi xây trên

các framework như LangGraph hay Mastra. Khi năng lực AI mới xuất hiện, kiến trúc agent dạng component mang lại các điểm tích hợp tự nhiên mà không phải tái cấu trúc toàn hệ thống. Bạn dễ dàng tích hợp tool mới vào các framework agent module hóa, và cập nhật cấu hình trung tâm để triển khai kỹ thuật prompt cải tiến cho toàn bộ agent.

Mở rộng năng lực với Agent Skills. Agent Skills là cách có cấu trúc để trang bị cho agent kiến thức chuyên môn, quy trình và tích hợp tool vượt ngoài năng lực gốc. Thay vì nhồi toàn bộ chuyên môn vào prompt, Skill đóng vai trò các gói năng lực dạng module mà agent có thể dùng khi cần.

Kiến trúc lắp ghép được: Các Skill có thể phối hợp trong tác vụ phức tạp và gọi Skill khác khi cần. Điều này mở ra những quy trình tinh vi - ví dụ, một skill tuân thủ có thể gọi skill phân tích tài liệu, skill này lại dùng một skill trích xuất chuyên biệt. Khả năng lắp ghép đó cho phép xây các tầng năng lực mà không tạo ra những khối triển khai nguyên khối.

Khi nào nên dùng Skill:

- Chuyên môn theo lĩnh vực (phân tích tài chính, rà soát pháp lý, nghiên cứu khoa học)
- Các quy trình chuẩn hóa tổ chức bạn đã tinh luyện
- Tích hợp tool chuyên biệt (cơ sở dữ liệu, API, hệ thống nội bộ)
- Thông lệ ngành và yêu cầu tuân thủ đặc thù

Cách triển khai: Skill tích hợp mượt với cả kiến trúc agent đơn lẫn đa agent. Trong hệ agent đơn, Skill mở rộng năng lực nền của agent. Trong hệ đa agent, mỗi agent có thể cấu hình bộ skill theo chuyên môn riêng - agent phân tích tài chính có thể dùng skill đánh giá rủi ro trong khi agent hỗ trợ khách hàng dùng skill tích hợp CRM.

Cách tiếp cận module này nghĩa là bạn cập nhật skill độc lập mà không phải viết lại logic agent, chia sẻ Skill giữa nhiều Agent, và mở rộng năng lực khi nhu cầu tổ chức tiến hóa.

Xây hệ thống quan sát được, tự giải thích chính mình. Ứng dụng AI thường vận hành như hộp đen, và agent còn chồng thêm các lớp phức tạp. Ngoài các thực hành chuẩn như structured logging, giám sát tập trung và distributed tracing, ứng dụng AI mang đến những thách thức observability đặc thù mà các công cụ giám sát hiệu năng truyền thống không được thiết kế để xử lý.

Vấn đề cốt lõi là hệ thống AI không tất định (non-deterministic), với quá trình suy luận khó nhìn thấu. Khi một AI agent lỗi hay hành xử bất thường, bạn không thể chỉ xem stack trace - bạn cần nhìn vào chuỗi prompt, đường ra quyết định của mô hình, ngữ cảnh truy xuất, mức tiêu thụ token và toàn bộ luồng suy luận. Cách debug truyền thống thường bó tay khi logic cốt lõi diễn ra bên trong một mạng nơ-ron.

Điểm nhận thức then chốt: debug ứng dụng AI đòi hỏi hiểu không chỉ chuyện gì đã xảy ra, mà cả vì sao mô hình ra quyết định đó và ngữ cảnh đã chảy qua chuỗi suy luận nhiều bước như thế nào.

Với các nguyên tắc nền tảng đã rõ, giờ hãy xem chúng áp vào từng mẫu kiến trúc cụ thể ra sao. Ta bắt đầu với cách tiếp cận đơn giản nhất nhưng hiệu quả cho đa số tình huống doanh nghiệp, rồi tiến dần đến những mẫu tinh vi hơn - những mẫu chứng minh được độ phức tạp của mình bằng mức tăng hiệu năng đo đếm được.

Hệ thống agent đơn

Trong hệ agent đơn, một agent chạy bằng AI hoạt động theo vòng lặp liên tục: quan sát môi trường, quyết định bước kế tiếp, và hành động để đạt mục tiêu.

Thông thường, tương tác với agent diễn ra theo khuôn này:

1. Người dùng giao cho agent một nhiệm vụ.

2. Agent lập kế hoạch, thực thi hành động dựa trên các tool sẵn có, quan sát kết quả và điều chỉnh cách tiếp cận theo phản hồi.

3. Agent lặp lại chu trình này đến khi hoàn thành nhiệm vụ hoặc chạm điều kiện dừng, ví dụ "tạm dừng tại đây chờ con người duyệt."

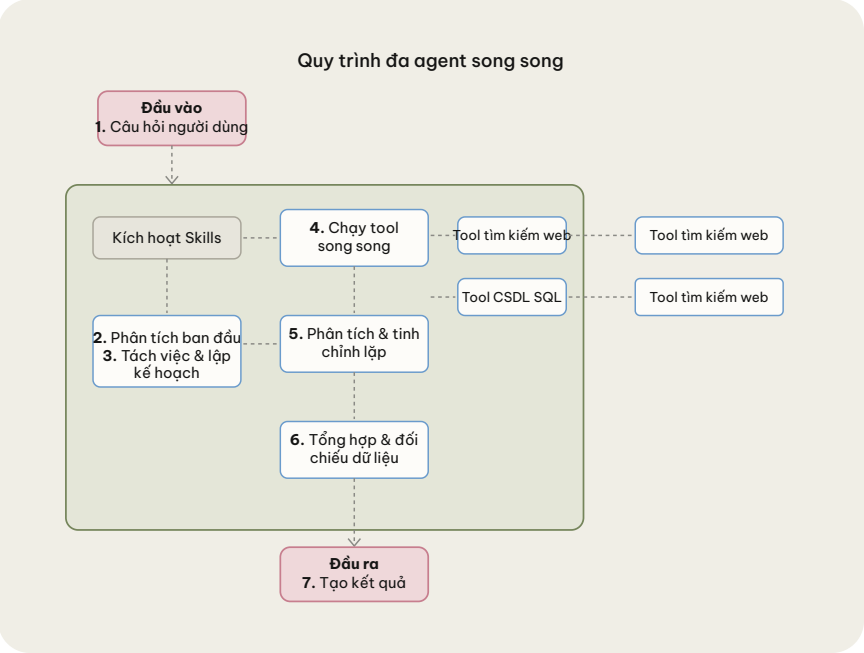
Tổng quan kiến trúc: Các thành phần lõi của một hệ agent đơn gồm: một mô hình AI đóng vai trò bộ máy suy luận, một prompt định nghĩa vai trò và năng lực của agent, và một bộ tool tích hợp cho phép agent tương tác với hệ thống bên ngoài và thực hiện các chức năng cụ thể. Skill bổ sung thêm một lớp năng lực, trang bị cho agent kiến thức chuyên ngành, quy trình và cách làm hay vượt ngoài nền huấn luyện gốc, giúp một agent đơn xử lý được những nhiệm vụ phức tạp lẽ ra cần nhiều agent chuyên biệt.

Khi nào nên dùng: Agent đơn tỏa sáng với các bài toán mở, nơi đường đi không rõ ngay từ đầu. Bạn không thể định sẵn lời giải vì không biết sẽ cần bao nhiêu bước hay chướng ngại nào sẽ xuất hiện.

Khi nào nên tránh: Khi bạn cần câu trả lời hoàn hảo ngay lần đầu, 100% số lần. Agent đơn là nguồn lực mạnh, nhưng để xử lý vấn đề phức tạp hay đạt độ chính xác cao nhất, bạn sẽ muốn dùng kiến trúc đa agent. Dù vậy, trước khi mở rộng lên hệ đa agent, hãy cân nhắc liệu thêm skill chuyên biệt cho agent đơn có đạt được yêu cầu chính xác một cách hiệu quả hơn không.

Ví dụ: Agent nghiên cứu đơn

Bạn triển khai một agent nghiên cứu được cấu hình với **Model Context Protocol (MCP)** để kết nối agent vào các hệ thống của bạn, gồm kho nội dung, công cụ kinh doanh và môi trường phát triển. Sơ đồ dưới đây cho thấy một agent đơn có thể xử lý nhiệm vụ tương đối phức tạp bằng nhiều tool ra sao.



1. Câu hỏi của người dùng:

Một nhân viên gửi cho agent nghiên cứu câu hỏi sau:
"Nghiên cứu các công cụ năng suất làm việc từ xa mà các đội kỹ sư đang dùng nhiều, và xem có công cụ nào tương quan với chỉ số năng suất nội bộ của chúng ta không"

2. Phân tích ban đầu:

Claude đánh giá câu hỏi:

- **Think:** *"Câu hỏi này cần hai nguồn dữ liệu tách biệt: nghiên cứu bên ngoài về công cụ năng suất và chỉ số nội bộ công ty. Mình nên tách thành các lượt tìm song song vì nghiên cứu bên ngoài ban đầu không phụ thuộc dữ liệu nội bộ. Cuối cùng mình sẽ cần đối chiếu tương quan các phát hiện."*

3. Skill vào việc: Agent nghiên cứu này tận dụng các skill chuyên biệt, gồm:

- Skill phương pháp nghiên cứu dẫn dắt cách rà soát tài liệu có hệ thống
- Skill tương quan dữ liệu cung cấp khung nhận diện các mẫu hình có ý nghĩa
- Skill business intelligence bảo đảm các phát hiện bám sát ưu tiên của tổ chức

Các Skill này giúp agent áp dụng những khung đã được kiểm chứng thay vì suy luận từ con số không, cải thiện cả độ chính xác lẫn hiệu quả.

4. Tách nhiệm vụ và lập kế hoạch:
Dựa trên phân tích, Claude lên kế hoạch tiếp cận:

- Tìm kiếm web bên ngoài về xu hướng sử dụng công cụ năng suất
- Truy vấn cơ sở dữ liệu nội bộ lấy chỉ số năng suất công ty
- Chạy tool song song để tối ưu thời gian
- Phương pháp đối chiếu tương quan để tổng hợp

5. Chạy tool song song:
Claude tận dụng khả năng gọi tool song song có sẵn để chạy nhiều tool cùng lúc:

- Tool tìm kiếm web (qua MCP): tìm các công cụ năng suất làm việc từ xa và dữ liệu mức độ sử dụng
- Tool cơ sở dữ liệu SQL (qua MCP): truy vấn chỉ số năng suất nội bộ theo đội và giai đoạn

Cả hai tool chạy đồng thời, giảm đáng kể tổng thời gian phản hồi.

6. Phân tích lặp và tinh chỉnh: Sau khi xử lý kết quả ban đầu, Claude dùng tool think để phân tích sâu hơn:

- **Think:** *"Tìm kiếm web trả về các nhóm công cụ khá đầy đủ và xu hướng chung, nhưng cần thêm dữ liệu cụ thể về lựa chọn của các đội kỹ sư. Truy vấn cơ sở dữ liệu thành công - đã có chỉ số năng suất nền. Cần tìm kiếm tinh chỉnh hơn về dữ liệu định lượng và phản hồi theo từng đội."*

Claude chạy các truy vấn bổ sung dựa trên phân tích này:

- Tool tìm kiếm web: tìm kiếm tinh chỉnh nhắm vào mẫu hình sử dụng riêng của giới kỹ sư
- Tool cơ sở dữ liệu SQL: truy vấn đối chiếu giai đoạn triển khai công cụ với biến động năng suất

7. Tổng hợp và đối chiếu dữ liệu: Dùng tool think để phân tích toàn diện:

Think: "Nghiên cứu bên ngoài cho thấy xu hướng sử dụng rõ rệt ở công cụ phát triển, nền tảng quản lý dự án và hệ thống liên lạc. Chỉ số nội bộ cho thấy năng suất biến thiên giữa các đội và các quý. Đang đối chiếu mốc thời gian triển khai với dữ liệu hiệu suất để tìm tương quan tiềm năng, có tính đến các yếu tố bên ngoài."

8. Tạo kết quả: Claude tổng hợp các phát hiện bằng **năng lực ngữ cảnh mở rộng** để giữ trọn ngữ cảnh hội thoại và đưa ra kết quả hợp nhất:

"Nghiên cứu xác định được một số nhóm công cụ năng suất làm việc từ xa đang được các đội kỹ sư dùng nhiều: công cụ môi trường phát triển..."

Hệ thống đa agent

Kiến trúc đa agent điều phối nhiều agent chuyên biệt để xử lý các bài toán phức tạp vượt quá năng lực của một hệ thống tổng quát đơn lẻ. Thay vì một mô hình AI ôm hết, nhiệm vụ được tách nhỏ, phân phối và thực thi trên nhiều agent, thường với chuyên môn riêng cho từng loại câu hỏi. Kết quả từ nhiều agent sau đó được tổng hợp thành một câu trả lời mạch lạc.

Nghiên cứu nội bộ của Anthropic cho thấy với các nhiệm vụ phức tạp đòi hỏi theo đuổi nhiều hướng độc lập cùng lúc, hệ đa agent vượt hệ agent đơn tới 90,2%. Điểm nhận thức then chốt: trí tuệ đạt đến một ngưỡng mà **"hệ đa agent trở thành con đường thiết yếu để nâng hiệu năng"** vì "nhóm agent làm được nhiều hơn hẳn" từng cá thể, hệt như tổ chức của con người.

Tổng quan kiến trúc: Nhiều agent với năng lực chuyên biệt cùng hướng tới mục tiêu chung. Có thể là agent điều phối giao việc cho các chuyên gia, hoặc cấu trúc phân cấp nơi agent cấp cao quản lý các subagent. Giao tiếp diễn ra trực tiếp giữa các agent hoặc qua bộ nhớ chung và hàng đợi thông điệp để phối hợp. Agent Skills cũng có thể được phân bổ chiến lược cho từng agent để tạo độ chuyên sâu.

Khi nào nên dùng: Hệ đa agent tỏa sáng khi agent đơn chạm giới hạn căn bản. Chọn kiến trúc đa agent khi: (1) nhiệm vụ là bài toán mở khó đoán trước các bước và cần độ linh hoạt để chuyển hướng hay **lần theo những mạch liên quan khi cuộc điều tra mở rộng**; (2) bạn cần chuyên môn sâu đến mức làm quá tải một agent tổng quát - nghiên cứu cho thấy **agent đơn rớt hiệu năng mạnh khi có từ hai lĩnh vực gây nhiễu trở lên**; hoặc (3) bài toán đòi hỏi truy vấn diện rộng **theo nhiều hướng độc lập cùng lúc**, nơi xử lý song song đem lại mức tăng hiệu năng đáng kể. Chúng đặc biệt hợp với nghiên cứu phức tạp, phân tích tổng hợp trải nhiều chuyên ngành, hay các kịch bản cần vận hành tự chủ kéo dài trên nhiều mảng tri thức.

Lưu ý triển khai: Hệ đa agent đem lại sức mạnh ấn tượng cho nhiệm vụ phức tạp, nhưng sức mạnh đó đi kèm độ phức tạp tương ứng cả về kiến trúc lẫn chi phí vận hành. Kiến trúc đa agent ngốn token rất nhanh, nên cần những nhiệm vụ mà giá trị kinh doanh xứng với chi phí hiệu năng tăng thêm. Hãy thiết kế hệ thống để liều lượng nỗ lực hợp lý - câu hỏi đơn giản không nên kích hoạt quy trình đa agent đắt đỏ.

Observability càng trở nên sống còn. Như đã nói, cách debug truyền thống thất bại vì agent ra quyết định động và không tất định giữa các lần chạy. Trong kiến trúc đa agent, nơi các quyết định của agent nhân lên, điều thiết yếu là triển khai tracing ghi nhận không chỉ hành vi từng agent mà cả mẫu hình ra quyết định và cấu trúc tương tác giữa các agent, để **chẩn đoán nguyên nhân gốc khi phối hợp trục trặc**. Thiếu observability toàn diện về cách agent giao tiếp, giao việc và tổng hợp kết quả, việc debug gần như bất khả thi khi hành vi phát sinh (emergent) xuất hiện từ các tương tác phức tạp.

Khi cân nhắc triển khai đa agent, hãy bắt đầu bằng việc xác định rõ mục tiêu và xây giải pháp đơn giản nhất đáp ứng yêu cầu. Thiết kế cho tính module và khả năng mở rộng ngay từ đầu; bạn sẽ biết ơn nền móng này khi cần thêm năng lực mới hay mở rộng năng lực hiện có.

Các mẫu kiến trúc

Hệ đa agent tổ chức quanh hai khái niệm phối hợp căn bản: kiến trúc tập trung và phi tập trung, mỗi loại giải các thách thức phối hợp và tình huống sử dụng khác nhau.

Hệ tập trung dùng mẫu phân cấp, nơi một giám sát trung tâm giao việc thông minh cho các agent chuyên biệt, tạo chuỗi trách nhiệm rõ ràng phản chiếu cấu trúc tổ chức hiệu quả của con người. Các hệ phân cấp này được gọi bằng nhiều tên như mẫu giám sát (supervisory), điều phối (orchestrator) hay định tuyến (router), mỗi tên là một biến thể hơi khác nhau của kiểm soát tập trung: có loại thiên về giao việc, loại thiên về quyết định định tuyến, loại khác lại điều phối trọn vẹn tương tác giữa các agent.

Hệ phi tập trung dùng mẫu cộng tác, nơi các agent tự chủ giao tiếp trực tiếp ngang hàng, thương lượng vai trò linh hoạt và giải bài toán phức tạp bằng trí tuệ phân tán. Hệ cộng tác đôi khi được gọi là kiến trúc bầy đàn (swarm) hay liên hợp (federated), phản ánh thiên hướng phối hợp tự phát thay vì kiểm soát áp đặt.

Đỡ bên dưới các mẫu kiến trúc này là **agentic workflow** - quy trình agent - cung cấp điều phối có cấu trúc cho các tiến trình nhiều bước, định nghĩa trình tự và điều kiện để agent thực thi nhiệm vụ trong môi trường phân tán.

Khác biệt cốt lõi nằm ở triết lý phối hợp: kiểm soát tập trung, tự chủ phân tán, hay điều phối có cấu trúc. Các tổ chức thường kết hợp những mẫu này để tạo giải pháp vững chắc, mở rộng được, khớp với yêu cầu kinh doanh riêng.

Hệ phân cấp/giám sát

Hệ phân cấp dùng một bộ điều khiển trung tâm để phối hợp nhiều agent theo vai trò thông qua giao việc thông minh. Agent giám sát phân tích yêu cầu đến, định tuyến tới đúng chuyên gia và tổng hợp phản hồi, tạo chuỗi trách nhiệm rõ ràng có thể mở rộng theo độ phức tạp của tổ chức.

Trong hệ phân cấp, từng **subagent** được xem như tool: agent giám sát dùng mô hình gọi tool để quyết định gọi "agent tool" nào. Mẫu này phản chiếu cách các đội nhóm hiệu quả vận hành: chuyên gia tập trung vào chuyên môn của mình trong khi điều phối viên lo phân việc và tích hợp. Subagent cũng có thể có subagent riêng, các nhóm này được trừu tượng hóa khỏi agent giám sát - giám sát chỉ làm việc với trưởng nhóm subagent và không biết về các tầng giao việc sâu hơn.

Bài toán kinh tế nghiêng về cách này dù tốn token hơn. Hệ đa agent tiêu thụ token nhiều hơn hẳn tương tác đơn, nhưng mức tăng hiệu năng xứng đáng với chi phí cho các nhiệm vụ giá trị cao, phức tạp, cần kiến thức chuyên sâu hoặc vượt giới hạn ngữ cảnh của agent đơn.

Các biến thể triển khai khác nhau ở cách cân bằng chi phí phối hợp với chất lượng phản hồi:

- Hệ điều phối trọn vẹn giữ toàn quyền giám sát tương tác người dùng và thực thi nhiệm vụ
- Hệ thiên định tuyến chuyên về quyết định giao việc, có thể chuyển hẳn việc giao tiếp với người dùng cho agent chuyên trách
- Hệ phối hợp lai chỉ kéo giám sát vào cuộc tùy độ phức tạp nhiệm vụ

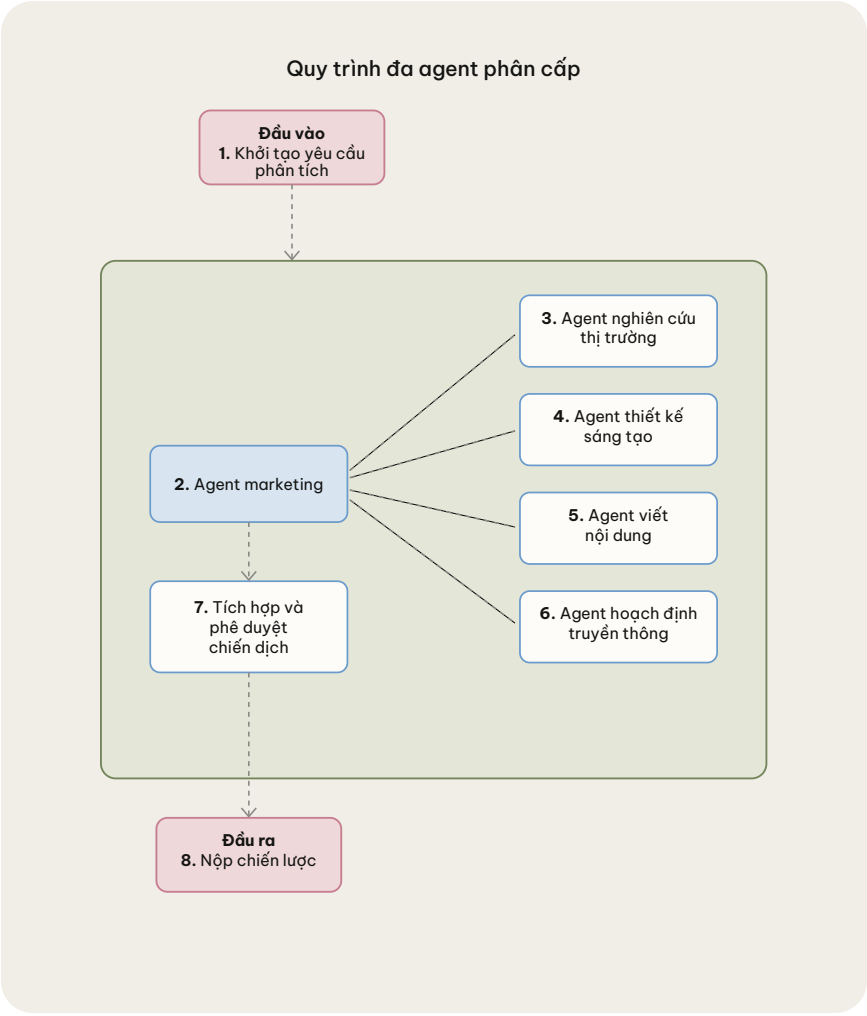
Thách thức chính: Quản lý ngữ cảnh

Agent điều phối có thể đối mặt bài toán căn bản: **ngữ cảnh phình to vượt khả năng quản lý hiệu quả của một agent**, tạo nghẽn hiệu năng khi agent chật vật giữ mạch nhất quán qua các tương tác kéo dài. Độ phức tạp ngữ cảnh này biểu hiện thành tràn cửa sổ ngữ cảnh, suy giảm chất lượng suy luận và lỗi phối hợp giữa các agent.

Triển khai thành công cần chiến lược quản lý ngữ cảnh vững: **context editing** tự động dọn các lệnh gọi tool và kết quả cũ khi bạn tiến gần giới hạn token mà vẫn giữ mạch hội thoại, còn memory tool cho agent lưu và truy lại thông tin bên ngoài cửa sổ ngữ cảnh qua hệ thống dạng file **bền vững qua các phiên**. Bạn cũng nên cân nhắc cho tool hỗ trợ phân trang, chọn khoảng, lọc và cắt bớt với mặc định hợp lý, giới hạn phản hồi ở mức quản lý được (cỡ 25.000 token) để **tránh cạn ngữ cảnh**.

Ví dụ: Quy trình đa agent phân cấp - Xây dựng chiến dịch marketing

Một agency marketing triển khai hệ đa agent phân cấp để xây các chiến dịch marketing toàn diện, với một agent giám sát điều phối các agent chuyên trách nhằm bảo đảm nhất quán chiến lược trong khi tận dụng chuyên môn sâu ở mọi cấu phần chiến dịch.



1. Nộp brief chiến dịch: Khách hàng gửi brief chiến dịch marketing gồm mục tiêu, đối tượng, giới hạn ngân sách, tiến độ và brand guideline vào hệ thống.

2. Agent giám đốc marketing (giám sát): Agent giám sát phân tích yêu cầu chiến dịch, xác định các hạng mục chính, định mức phân bổ nguồn lực và lập kế hoạch thực thi chiến lược ánh xạ từng nhiệm vụ cụ thể tới đúng agent chuyên trách.

3. Agent nghiên cứu thị trường: Nhận chỉ đạo từ giám sát để phân tích đối tượng mục tiêu, nghiên cứu bức tranh cạnh tranh và đánh giá cơ hội thị trường, báo cáo kết quả về agent giám đốc marketing.

4. Agent thiết kế sáng tạo: Được giám sát giao phát triển concept hình ảnh, tài sản thương hiệu và khung thiết kế dựa trên phát hiện nghiên cứu thị trường và brand guideline, với giám sát duyệt định hướng sáng tạo.

5. Agent viết nội dung: Nhận nhiệm vụ từ giám sát để xây chiến lược thông điệp, lời quảng cáo và nội dung trên mọi kênh, bảo đảm nhất quán với định hướng sáng tạo và định vị thị trường các agent trước đã chốt.

6. Agent hoạch định truyền thông: Được giám sát chỉ đạo xây khuyến nghị tổ hợp kênh, lựa chọn kênh, phân bổ ngân sách giữa các nền tảng và chiến lược thời điểm dựa trên hiểu biết về đối tượng và yêu cầu sáng tạo.

7. Tích hợp và phê duyệt chiến dịch: Agent giám đốc marketing tổng hợp toàn bộ đầu ra chuyên trách, bảo đảm mạch lạc chiến lược, xử lý xung đột giữa các khuyến nghị và hoàn thiện đề xuất chiến dịch tích hợp.

8. Nộp chiến lược chiến dịch hoàn chỉnh: Chiến dịch marketing tích hợp cuối cùng với tài sản sáng tạo, kế hoạch truyền thông, phân bổ ngân sách, tiến độ và chỉ số thành công được bàn giao cho khách hàng.

Hệ cộng tác

Hệ cộng tác cho phép nhiều agent chuyên biệt làm việc cùng nhau theo thời gian thực qua các cơ chế phối hợp tinh vi, chia sẻ thông tin và phối hợp hành động để đạt kết quả vượt năng lực từng agent. Khác hệ phân cấp có kiểm soát trung tâm, mẫu cộng tác đề cao tương tác ngang hàng, nơi agent giao tiếp

trực tiếp, thương lượng vai trò linh hoạt và cùng nhau giải bài toán phức tạp bằng trí tuệ phân tán.

Trong hệ cộng tác, agent vận hành như những thực thể tự chủ: giao tiếp, phối hợp và cộng tác qua nhiều cơ chế để đạt mục tiêu chung. Cách này phản chiếu teamwork của con người, nơi chuyên gia đóng góp chuyên môn trong khi vẫn ý thức về mục tiêu lớn. Điểm khác biệt then chốt: sự phối hợp nảy sinh từ tương tác giữa các agent thay vì bị áp từ một trung tâm quyền lực.

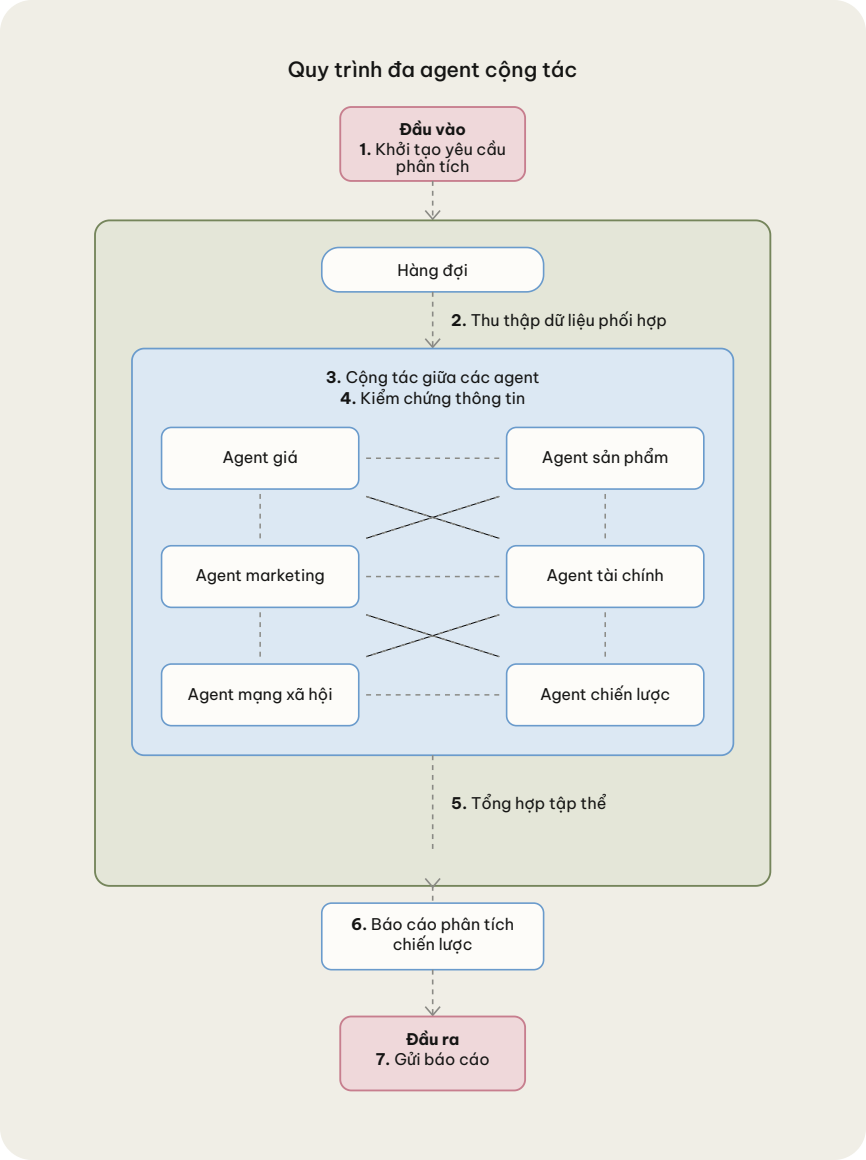
Các biến thể triển khai khác nhau ở cách agent phối hợp và chia sẻ ngữ cảnh:

- Điều phối group chat cho nhiều agent giải bài toán, ra quyết định hay kiểm tra chéo công việc bằng cách tham gia một luồng hội thoại chung, cộng tác qua thảo luận
- Phối hợp hướng sự kiện dùng sự kiện như ngôn ngữ chung, đóng vai trò các bản cập nhật có cấu trúc giúp agent diễn giải chỉ dẫn, chia sẻ ngữ cảnh và phối hợp nhiệm vụ
- Kiến trúc blackboard cung cấp kho tri thức chung nơi mọi agent đọc và ghi vào một kho trung tâm đóng vai trò trí nhớ tập thể

Thách thức chính: Độ phức tạp giao tiếp và hành vi phát sinh khó lường

Hệ cộng tác đối mặt thách thức căn bản trong quản lý giao tiếp giữa agent và dự đoán hành vi hệ thống. Giao tiếp dày đặc giữa các agent kéo theo chi phí tính toán và độ phức tạp tăng, trong khi hệ đa agent có những hành vi phát sinh không được lập trình cụ thể, nơi **thay đổi nhỏ có thể ảnh hưởng khó lường đến cách agent hành xử**. Thành công đòi hỏi khung cộng tác định nghĩa phân công lao động, cách tiếp cận giải quyết vấn đề và ngân sách nỗ lực thay vì chỉ dẫn cứng. Thách thức bổ sung gồm việc ngăn agent đùn đẩy nhiệm vụ vô hạn và xây cơ chế xử lý xung đột vững chắc.

Ví dụ: Quy trình đa agent cộng tác - Thu thập thông tin cạnh tranh



Một hãng tư vấn chiến lược triển khai hệ thống thu thập thông tin đa agent cộng tác, nơi các agent phân tích chuyên trách làm việc cùng nhau theo thời gian thực, đối chiếu chéo phát hiện và dựng bức tranh cạnh tranh toàn cảnh vượt năng lực từng agent nhờ trí tuệ tập thể.

1. Khởi tạo yêu cầu: Khách hàng yêu cầu phân tích cạnh tranh toàn diện, kích hoạt thu thập thông tin phối hợp trên toàn bộ agent phân tích chuyên trách.

2. Thu thập dữ liệu phối hợp: Yêu cầu của khách được đưa vào hàng đợi. Các agent phân tích giá, sản phẩm, marketing, tài chính, mạng xã hội và chiến lược thiết lập kênh liên lạc và phân chia trách nhiệm theo dõi để tránh trùng lặp.

3. Cộng tác giữa các agent: Các agent liên tục chia sẻ phát hiện theo thời gian thực - agent giá báo cho agent sản phẩm về tương quan tính năng-giá, agent marketing chia sẻ dữ liệu chiến dịch với agent tài chính, agent mạng xã hội cung cấp tín hiệu cảm xúc thị trường cho agent chiến lược.

4. Kiểm chứng thông tin: Tất cả agent đối chiếu chéo các phát hiện để tìm mâu thuẫn, xác thực thông tin qua nhiều nguồn dữ liệu và dựng hồ sơ đối thủ đã được kiểm chứng.

5. Tổng hợp tập thể: Các agent cộng tác tích hợp những phát hiện đa chiều, đánh giá cơ hội thị trường và phát triển dự báo về nước đi chiến lược của đối thủ.

6. Báo cáo phân tích chiến lược: Agent báo cáo xây bản phân tích bức tranh cạnh tranh toàn diện với phát hiện đã kiểm chứng, dự báo và khuyến nghị chiến lược dựa trên trí tuệ tập thể của các agent.

7. Gửi báo cáo: Báo cáo phân tích tích hợp cuối cùng kèm số liệu được bàn giao cho khách hàng.

Quy trình agent (agentic workflow)

Agentic workflow định nghĩa cấu trúc vận hành của agent: cách chúng giao tiếp, bàn giao nhiệm vụ và cộng tác hướng tới mục tiêu chung. Khác với hành vi động của từng agent, workflow được định nghĩa trước và tĩnh. Hai mẫu quy trình agent phổ biến là tuần tự và phân cấp.

Quy trình tuần tự

Quy trình tuần tự dùng luồng điều khiển định trước với đường thực thi xác định, bảo đảm các bước chuyển giữa agent dễ đoán - lý tưởng cho những tiến trình lặp lại như chuỗi phê duyệt tài liệu hay kiểm tra tuân thủ. Các quy trình này cho dấu vết kiểm toán rõ ràng và hành vi tất định, rất hợp môi trường có quản chế nơi tính nhất quán và truy vết được của quy trình là sống còn.

Quy trình tuần tự có thể dùng điểm quyết định do phần mềm định nghĩa, như logic điều kiện theo kết quả nhiệm vụ hay thay đổi trạng thái hệ thống, hoặc định tuyến bằng AI, nơi mô hình quyết định luồng điều khiển dựa trên kết quả trung gian và yếu tố ngữ cảnh. Cách lai này vừa giữ độ tin cậy của đường đi định sẵn, vừa linh hoạt thích ứng theo phân tích nội dung hay điều kiện động.

Lợi thế then chốt là khả năng dự đoán vận hành: bạn vạch được toàn bộ luồng, ước tính chi phí thực thi và debug bằng cách soi từng chặng cụ thể. Nhưng sự dễ đoán đó đánh đổi bằng độ linh hoạt khi gặp trường hợp biên hay kịch bản mới không khớp cấu trúc định sẵn.

Khi nào nên dùng: Dùng quy trình tuần tự khi nhiệm vụ tách gọn được thành các phần việc cố định. Mục tiêu chính là đánh đổi độ trễ lấy độ chính xác cao hơn, bằng cách biến mỗi lượt gọi AI thành một việc dễ hơn, tập trung hơn.

Cân nhắc điều phối tuần tự trong các kịch bản: tiến trình nhiều chặng có phụ thuộc tuyến tính rõ và diễn tiến dễ đoán, pipeline biến đổi dữ liệu nơi mỗi chặng bồi thêm giá trị cho chặng sau, các chặng không song song hóa được, và yêu cầu tinh chỉnh

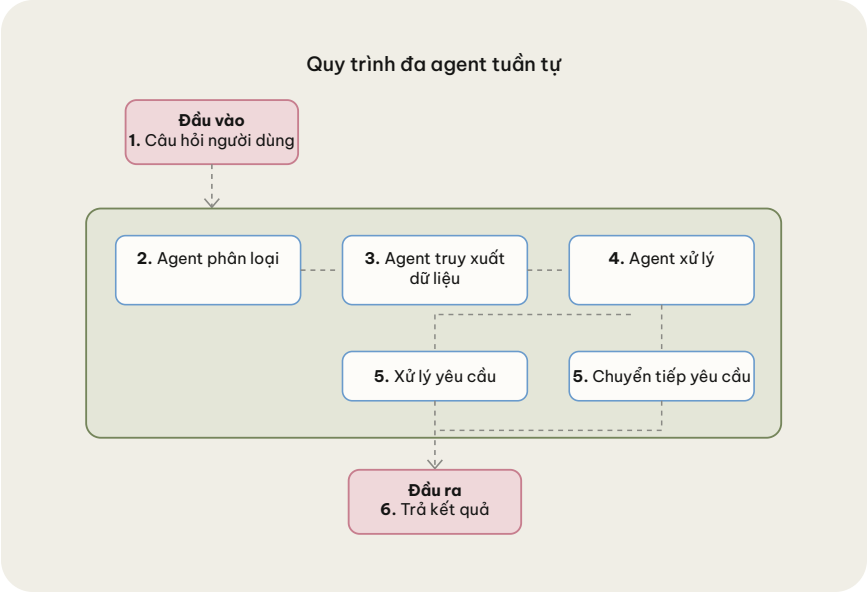
lũy tiến kiểu soạn-duyệt-trau chuốt.

Dùng mẫu tuần tự khi bạn nắm rõ độ sẵn sàng và đặc tính hiệu năng của từng agent trong pipeline, và khi lỗi hay chậm trễ ở một agent vẫn chấp nhận được với tổng thể nhiệm vụ.

Ví dụ nơi quy trình tuần tự phát huy gồm: tạo nội dung marketing rồi dịch sang nhiều ngôn ngữ; viết dàn ý tài liệu, kiểm tra dàn ý đạt tiêu chí cụ thể, rồi **viết bản đầy đủ dựa trên dàn ý đã duyệt**.

Khi nào nên tránh: Tránh quy trình tuần tự cho các tiến trình chỉ gồm vài chặng mà một agent làm tốt được, khi agent cần cộng tác thay vì bàn giao, hay khi quy trình cần quay lui hoặc lặp.

Ví dụ: Quy trình đa agent tuần tự - tự động hóa phân tích dữ liệu



Một công ty triển khai giải pháp quy trình đa agent để tự động hóa các yêu cầu phân tích dữ liệu, tạo insight nhanh mà không dồn nghẽn lên đội data science.

1. Yêu cầu phân tích: Một bên liên quan gửi yêu cầu phân tích dữ liệu qua hệ thống (vd. "Phân tích hiệu quả bán hàng Q4 theo vùng" hay "Tìm các yếu tố rủi ro khiến khách rời bỏ").

2. Agent định phạm vi: Agent định phạm vi phân tích yêu cầu đến, xác định loại phân tích (mô tả, chẩn đoán, dự báo hay khuyến nghị), nhận diện nguồn dữ liệu và phương pháp cần thiết, đánh giá độ phức tạp và định tuyến vào đúng nhánh phân tích.

3. Agent kỹ thuật dữ liệu: Agent kỹ thuật dữ liệu dùng kết quả định phạm vi để trích dữ liệu từ các nguồn liên quan (kho dữ liệu, API, cơ sở dữ liệu), làm sạch và kiểm tra dữ liệu, xử lý giá trị thiếu và ngoại lai, tạo các đặc trưng phù hợp và chuẩn bị bộ dữ liệu sẵn sàng phân tích.

4. Agent phân tích: Agent phân tích nhận dữ liệu đã chuẩn bị và chạy quy trình phân tích phù hợp - chạy kiểm định thống kê, dựng mô hình, tạo biểu đồ, nhận diện mẫu hình và insight chính - hoặc gắn cờ các yêu cầu phức tạp cần nhà khoa học dữ liệu vào cuộc kèm gói bàn giao chi tiết.

5. Duyệt/leo thang: Kết quả phân tích hoặc được tự động kiểm tra và duyệt phân phối, hoặc xếp hàng chờ một nhà khoa học dữ liệu cấp cao rà soát chất lượng và tinh chỉnh diễn giải.

6. Bàn giao insight: Đầu ra cuối (báo cáo, dashboard, dự báo của mô hình hay khuyến nghị) được đóng gói và gửi tới các bên liên quan qua kênh họ ưa dùng.

Quy trình song song

Quy trình song song phân phối các nhiệm vụ độc lập cho nhiều agent chạy cùng lúc, kết quả được gộp hoặc xử lý đồng thời. Mẫu này tỏa sáng khi nhiệm vụ cần góc nhìn hay chuyên môn đa dạng, đem lại mức tăng tốc đáng kể nhờ xử lý đồng thời.

Mẫu điều phối đồng thời chạy nhiều agent cùng lúc trên cùng một nhiệm vụ, để mỗi agent đưa ra phân tích độc lập từ góc nhìn hay chuyên môn riêng. Cách này giống mẫu thiết kế cloud **fan-out/fan-in**, nơi kết quả thường được gộp lại nhưng không bắt buộc.

Mẫu này giải các kịch bản cần insight hay cách tiếp cận đa dạng cho cùng một bài toán. Thay vì xử lý tuần tự, mọi agent chạy song song, giảm tổng thời gian và phủ toàn diện không gian bài toán. Mỗi agent có thể độc lập tạo kết quả trong phần việc của mình, như gọi tool hay cập nhật các kho dữ liệu khác nhau.

Agent vận hành độc lập, không bàn giao kết quả cho nhau, dù một agent có thể gọi thêm agent khác bằng cách điều phối riêng. Mẫu này hỗ trợ cả gọi tất định tới mọi agent đã đăng ký lẫn lựa chọn động theo yêu cầu nhiệm vụ.

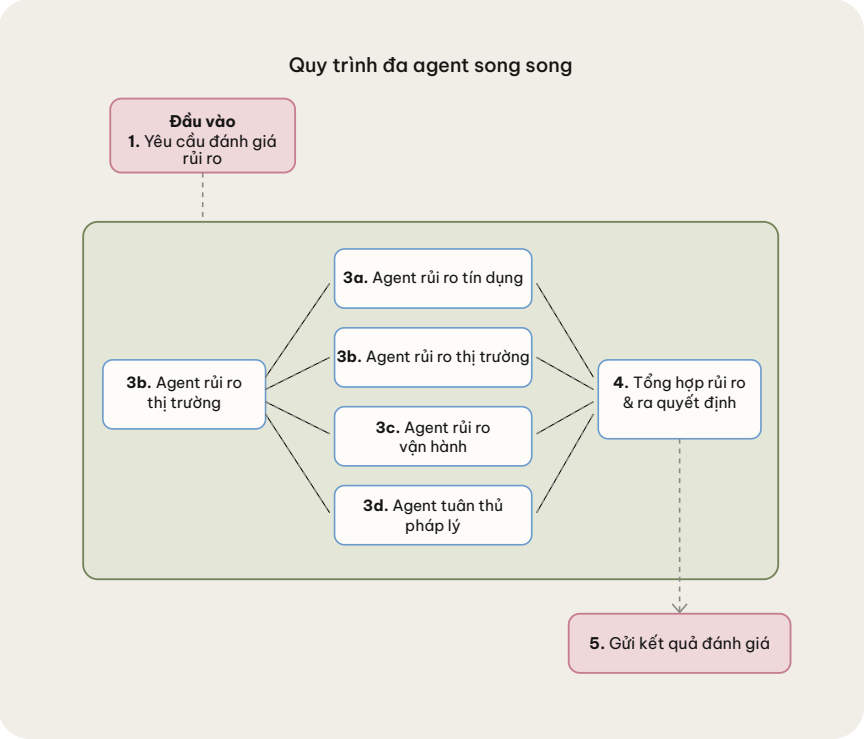
Khi nào nên dùng: Dùng song song hóa khi các phần việc tách được có thể xử lý cùng lúc để tăng tốc, hoặc khi cần nhiều góc nhìn để kết quả đáng tin hơn. Với nhiệm vụ phức tạp nhiều khía cạnh, mô hình AI thường làm tốt hơn khi mỗi khía cạnh được một lượt gọi riêng xử lý, cho phép tập trung trọn vẹn vào từng khía cạnh cụ thể.

Ví dụ nơi song song hóa hữu ích gồm cách chia phần (sectioning) như dựng guardrail, nơi một mô hình xử lý câu hỏi người dùng còn mô hình khác sàng lọc nội dung không phù hợp; hay **tự động hóa đánh giá**, nơi mỗi lượt gọi đánh giá một khía cạnh hiệu năng của mô hình. Mẫu bỏ phiếu (voting) hợp với rà soát lỗ hổng code bằng nhiều prompt khác nhau, hay **đánh giá độ phù hợp nội dung với nhiều prompt** dùng ngưỡng phiếu khác nhau để cân bằng dương tính giả và âm tính giả.

Khi nào nên tránh: Tránh quy trình song song khi agent cần xây tiếp trên công việc của nhau hoặc cần ngữ cảnh tích lũy theo trình tự cụ thể, khi nhiệm vụ đòi hỏi thứ tự thao tác nhất định hay kết quả tất định từ việc chạy theo trình tự định sẵn, hoặc khi giới hạn tài nguyên như hạn mức mô hình khiến xử lý song song kém hiệu quả. Đừng dùng mẫu song song khi agent không thể phối hợp tin cậy các thay đổi lên trạng thái chung hay hệ thống bên ngoài trong lúc chạy đồng thời, khi chưa có chiến lược xử lý xung đột rõ ràng cho kết quả mâu thuẫn giữa các agent, hoặc khi logic gộp kết quả quá phức tạp hay làm giảm chất lượng đầu ra.

Ví dụ: Quy trình đa agent song song - đánh giá rủi ro tài chính

Một định chế tài chính triển khai quy trình đa agent song song để thẩm định hồ sơ vay và cơ hội đầu tư, ra quyết định nhanh hơn trong khi vẫn phân tích rủi ro toàn diện trên các chiều then chốt.



1. Yêu cầu đánh giá rủi ro: Hồ sơ vay hoặc đề xuất đầu tư được gửi vào hệ thống để thẩm định rủi ro toàn diện.

2. Agent gom dữ liệu: Agent gom dữ liệu thu thập mọi thông tin liên quan gồm báo cáo tín dụng, báo cáo tài chính, dữ liệu thị trường, hồ sơ pháp lý và chỉ số hiệu quả lịch sử từ các nguồn trong và ngoài.

3. Các agent song song (dùng tool riêng của mình)

3a. Agent rủi ro tín dụng: Đồng thời phân tích uy tín tín dụng của bên vay, tỷ lệ nợ trên thu nhập, lịch sử trả nợ và chất lượng tài sản bảo đảm để tính điểm rủi ro tín dụng và xác suất vỡ nợ.

3b. Agent rủi ro thị trường: Song song đánh giá biến động thị trường, độ nhạy lãi suất, mức phơi nhiễm theo ngành và các chỉ báo kinh tế để ước tính tổn thất tiềm năng từ biến động thị trường và suy thoái.

3c. Agent rủi ro vận hành: Cùng lúc soi các rủi ro quy trình nội bộ, dấu hiệu gian lận, lỗ hổng tuân thủ và năng lực vận hành xử lý giao dịch, nhận diện các điểm hỏng hóc hay bất thường tiềm tàng.

3d. Agent tuân thủ pháp lý: Đồng thời rà yêu cầu pháp lý, kiểm tra phòng chống rửa tiền, tuân thủ định danh khách hàng (KYC) và các giới hạn theo từng khu vực tài phán để bảo đảm tuân thủ trọn vẹn.

4. Bộ máy tổng hợp rủi ro và ra quyết định: Toàn bộ đánh giá rủi ro song song được hợp nhất, đặt trọng số theo chính sách của tổ chức và tổng hợp thành hồ sơ rủi ro toàn diện kèm khuyến nghị hành động.

5. Gửi kết quả đánh giá rủi ro: Bản thẩm định rủi ro đa agent cuối cùng với khuyến nghị duyệt/từ chối, điểm rủi ro và báo cáo phân tích chi tiết được chuyển tới người ra quyết định.

Đánh giá - tối ưu (evaluator-optimizer)

Quy trình evaluator-optimizer dùng hai hệ AI theo chu kỳ lặp: một bên tạo nội dung, bên kia đánh giá và phản hồi, lặp đến khi đạt chuẩn chất lượng. Mẫu này đem lại cải thiện đáng kể khi triển khai đúng, dù chi phí token cao hơn.

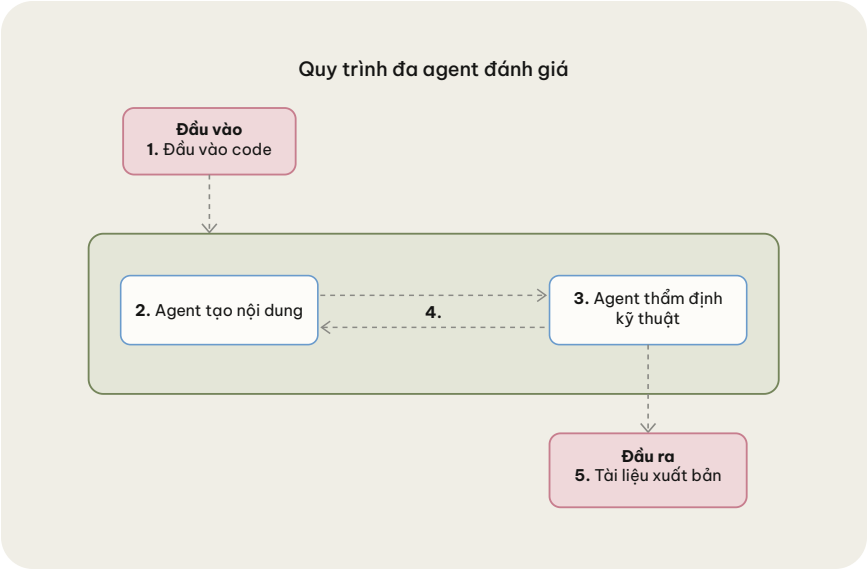
Mẫu vận hành qua vòng phản hồi có cấu trúc: bộ tạo (generator) sinh phản hồi ban đầu và tiếp nhận góp ý để cải thiện dần, còn bộ đánh giá (evaluator) **chấm nội dung theo tiêu chí định sẵn** và đưa chỉ dẫn hành động được. Cách này giống quan hệ tác giả - biên tập viên, với các gợi ý cụ thể được đưa vào bản thảo sửa.

Khi nào nên dùng: Dùng quy trình evaluator-optimizer khi tồn tại tiêu chí đánh giá rõ và việc tinh chỉnh lặp mang lại giá trị chứng minh được qua vòng phản hồi AI. Mẫu này tỏa sáng với sáng tạo nội dung cần độ tinh tế như dịch văn chương, sinh code có yêu cầu bảo mật, văn bản giao tiếp chuyên nghiệp nơi giọng điệu quan trọng, và các nhiệm vụ nghiên cứu cần suy luận nhiều bước có kiểm chứng.

Khi nào nên tránh: Tránh quy trình evaluator-optimizer khi chất lượng ngay lần đầu đã đạt yêu cầu, tiêu chí đánh giá chủ quan hay mơ hồ, hoặc khi ràng buộc thời gian và chi phí lấn át lợi ích chất lượng. Đừng dùng mẫu này cho ứng dụng thời gian thực cần phản hồi tức thì, các việc thường nhật đơn giản như phân loại cơ bản, hay môi trường eo hẹp tài nguyên với ngân sách token chặt. Tránh khi đã có lời giải tất định, khi bộ đánh giá thiếu chuyên môn lĩnh vực để góp ý có giá trị, hoặc khi mức suy giảm hiệu năng vượt quá lợi ích.

Ví dụ: Quy trình đa agent đánh giá - Tạo tài liệu API

Một tổ chức phát triển phần mềm triển khai quy trình evaluator-optimizer để tự động sinh tài liệu API đầy đủ từ codebase, bảo đảm chính xác kỹ thuật và thân thiện với lập trình viên qua các chu kỳ tinh chỉnh lặp giúp xóa nghẽn tài liệu thủ công.



1. Đầu vào code: Đội phát triển nạp codebase API vào hệ thống sinh tài liệu.

2. Agent tạo nội dung: Phân tích codebase và soạn bản tài liệu đầu tiên gồm mô tả endpoint, tham số, ví dụ và yêu cầu xác thực.

3. Agent thẩm định kỹ thuật: Đối chiếu độ chính xác của tài liệu với phần code triển khai thực tế, kiểm tra kiểu tham số, độ phủ endpoint và tính đúng của ví dụ.

4. Chu kỳ tinh chỉnh: Bộ tạo tiếp nhận góp ý từ cả hai bộ đánh giá và cải thiện tài liệu qua từng vòng đến khi đạt mọi tiêu chí.

5. Tài liệu xuất bản: Tài liệu API hoàn thiện được tự động đăng lên cổng lập trình viên kèm ví dụ tương tác và tư liệu tham chiếu đầy đủ.

Tiến trình này thường chạy 2-4 vòng, cải thiện đáng kể chất lượng tài liệu trong khi giữ độ chính xác kỹ thuật.

Các mẫu mới nổi

Khi các tổ chức đẩy xa giới hạn của AI agent, một số mẫu thử nghiệm đang đi từ phòng nghiên cứu vào những ca triển khai giai đoạn đầu. Phần dưới đây điểm qua vài mẫu mới nổi như vậy.

Mẫu agent: Sinh agent động

Sinh agent động là hướng thử nghiệm đang chớm nở, đưa tính module đến tận cùng logic của nó: agent được tạo ngay lúc chạy bằng cách lắp các thành phần từ thư viện prompt, tool và cấu hình, rồi giải tán khi xong nhiệm vụ. Dù chưa hệ thống production nào triển khai khả năng tạo động đúng nghĩa, nền móng kỹ thuật đã tồn tại rải rác trong nhiều dự án nghiên cứu và framework thử nghiệm như **AutoGen** hay **Semantic Kernel**.

Mẫu này hứa hẹn lợi thế hấp dẫn về tối ưu tài nguyên và hiệu năng theo từng nhiệm vụ; thay vì duy trì các agent cấu hình sẵn, hệ thống có thể phân tích yêu cầu đến và tự khởi tạo agent với đúng năng lực cần thiết, rồi giải phóng tài nguyên khi nhiệm vụ hoàn tất. Tuy nhiên, các thách thức lớn vẫn còn quanh độ phức tạp quản lý ngữ cảnh, rủi ro hành vi phát sinh và chi phí phụ trội của việc tạo động. Nghiên cứu hiện tại về phối hợp đa agent và kiến trúc hướng sự kiện đang đặt nền móng, nhưng các tổ chức nên xem đây là vùng đất thử nghiệm.

Mẫu kiến trúc: Hệ mạng lưới/ngang hàng

Kiến trúc mạng lưới là bước tiến hóa đáng kể trong phối hợp đa agent, xóa nghẽn phân cấp nhờ "giao tiếp nhiều-nhiều, nơi agent nào cũng nói chuyện trực tiếp được với agent khác". Đo đạc ban đầu cho thấy "kiến trúc bầy đàn nhỉnh hơn kiến trúc giám sát trên mọi mặt" vì agent cộng tác trực tiếp không qua tầng phiên dịch giám sát.

Khung quyết định: Mẫu nào cho tình huống nào

Hiểu các mẫu kiến trúc mới chỉ là bước đầu. Thách thức cốt yếu với lãnh đạo kỹ thuật là chọn đúng cách tiếp cận cho các ràng buộc cụ thể của bạn: ngân sách, tiến độ, độ phức tạp và khẩu vị rủi ro. Thay vì chọn theo độ tinh vi kỹ thuật, các ca triển khai thành công ghép độ phức tạp kiến trúc với giá trị kinh doanh qua việc đánh giá có hệ thống ba chiều then chốt.

Ba câu hỏi then chốt

Trước khi đi vào từng mẫu, mọi đội doanh nghiệp cần trả lời các câu hỏi nền tảng:

1. Bạn cần mức kiểm soát nào?

Yêu cầu kiểm soát cao (tuân thủ pháp lý, giao dịch tài chính, vận hành trọng yếu về an toàn) → Bắt đầu với agent đơn hoặc quy trình tuần tự

Hãy nghĩ thế này: nếu bạn phải giải thích chính xác vì sao hệ thống ra một quyết định cụ thể cho kiểm toán viên, cơ quan quản lý hay ban điều hành, bạn muốn hành vi dễ đoán, truy vết được. Một agent đơn duyệt hồ sơ vay với tiêu chí quyết định rõ ràng dễ kiểm toán hơn hẳn một hệ đa agent nơi ba mô hình AI khác nhau cùng góp vào khuyến nghị.

Yêu cầu kiểm soát vừa (hỗ trợ khách hàng, sáng tạo nội dung, phân tích dữ liệu) → Cân nhắc hệ đa agent phân cấp

Với các kịch bản cần linh hoạt nhưng vẫn muốn giám sát, hệ phân cấp cho bạn cả hai: agent giám sát giữ luật kinh doanh trong khi agent chuyên trách xử lý phần phức tạp.

Yêu cầu kiểm soát thấp (nghiên cứu, brainstorm, phân tích phức tạp) → Hệ đa agent cộng tác trở thành lựa chọn khả thi

Khi mục tiêu là khám phá khả năng hay xử lý bài toán thực sự phức tạp, tính khó lường của agent cộng tác trở thành tính năng, không phải lỗi.

2. Lĩnh vực bài toán của bạn phức tạp đến đâu?

Bài toán một lĩnh vực (trả lời câu hỏi sản phẩm, xử lý đổi trả, tạo báo cáo) → Agent đơn xử lý hiệu quả

Đừng thiết kế thừa. Nếu công việc gồm các tác vụ đơn giản, lặp lại, một agent đơn thiết kế tốt nhiều khả năng là đủ.

Bài toán đa lĩnh vực nhưng dễ đoán (onboarding nhân viên, quy trình tuân thủ, phân tích chuẩn) → Quy trình tuần tự hoặc song song

Khi bạn vạch được các bước nhưng cần chuyên môn khác nhau ở mỗi chặng, workflow cho cấu trúc mà không quá phức tạp.

Bài toán mở, phức tạp (phân tích chiến lược, dự án nghiên cứu, chẩn đoán hệ thống) → Kiến trúc đa agent

Những bài toán này cần tách nhỏ và nhiều cách tiếp cận. Nếu công việc hưởng lợi từ nhiều góc nhìn hay **Skill chuyên biệt**, hệ đa agent có thể hợp lý.

3. Ràng buộc tài nguyên của bạn là gì?

Ngân sách/token hạn chế → Agent đơn hoặc quy trình song song thiết kế kỹ

Hệ đa agent dùng nhiều token gấp khoảng 10-15 lần agent đơn. Hãy tính toán trên khối lượng dự kiến trước khi cam kết với kiến trúc phức tạp.

Áp lực ra thị trường nhanh → Bắt đầu với agent đơn, vạch lộ trình tiến hóa

Bạn triển khai được một agent đơn trong vài tuần. Hệ đa agent cần nhiều tháng mới chỉn chu. Hãy xây thứ chạy được trước, rồi nâng cấp.

Sáng kiến chiến lược dài hạn → Thiết kế cho tiến hóa module

Nếu đây là sáng kiến nhiều năm, hãy xây agent đơn đầu tiên với các interface hỗ trợ thêm agent về sau. Thiết kế cho tiến hóa ngay từ đầu: giữ trải nghiệm người dùng nhất quán trong khi chuẩn bị năng lực thay đổi kiến trúc backend khi yêu cầu lớn lên.

4. Bạn có cần chuyên môn lĩnh vực sâu?

Một lĩnh vực với quy trình đã định hình → Agent đơn kèm Skill chuyên biệt

Trước khi nhảy sang kiến trúc đa agent, hãy cân nhắc liệu một agent đơn trang bị skill chuyên ngành có giải được bài toán không. Skill cho chuyên môn sâu mà không kéo theo độ phức tạp của phối hợp đa agent.

Nhiều lĩnh vực tách bạch cần phối hợp → Hệ đa agent với Skill chuyên biệt

Khi các lĩnh vực phải phối hợp (vd. rà soát pháp lý phối hợp với phân tích tài chính), hệ đa agent nơi mỗi agent có bộ Skill phù hợp cho cả chuyên môn hóa lẫn phối hợp.

Ví dụ: Một hệ rà soát hợp đồng có thể khởi đầu là agent đơn dùng skill pháp lý. Khi độ phức tạp tăng, nó có thể tiến hóa thành hệ đa agent với các agent riêng cho phân tích hợp đồng, đánh giá rủi ro và kiểm tra tuân thủ - mỗi agent có bộ Skill chuyên biệt của mình.

Hướng dẫn chọn mẫu

Các ràng buộc trên quy ra khuyến nghị kiến trúc rõ ràng:

Agent đơn hợp nhất với:

- Quy trình chăm sóc khách hàng cho các nhóm sản phẩm được định nghĩa rõ
- Xử lý tài liệu với luật kinh doanh rõ ràng
- Review code và các tác vụ phát triển cơ bản
- Phân tích và báo cáo thường nhật

Quy trình tuần tự hợp nhất với:

- Quy trình phê duyệt nhiều bước
- Pipeline sản xuất nội dung (soạn → duyệt → đăng)
- Biến đổi và kiểm tra dữ liệu
- Kiểm tra tuân thủ theo nhiều tiêu chí

Quy trình song song hợp nhất với:

- Nhiều góc nhìn giúp nâng chất lượng
- Các phân tích độc lập chạy đồng thời được
- Tốc độ quan trọng hơn chi phí phối hợp
- Đánh giá rủi ro cần góc nhìn đa dạng

Hệ đa agent hợp nhất với:

- Giải bài toán phức tạp cần chuyên môn đa dạng
- Dự án nghiên cứu và phân tích
- Tương tác khách hàng động trải nhiều hệ thống
- Hoạch định chiến lược và hỗ trợ quyết định

Tiến hóa thực tế: hành trình của một nền tảng thương mại điện tử

- **Giai đoạn 1:** Agent đơn cho câu hỏi khách hàng (chứng minh giá trị)
- **Giai đoạn 2:** Mẫu định tuyến tách trạng thái đơn hàng, câu hỏi sản phẩm, khiếu nại
- **Giai đoạn 3:** Agent chuyên trách cho từng nhóm với ngữ cảnh chung

- **Giai đoạn 4:** Hệ đa agent với phối hợp tồn kho, thanh toán và vận chuyển
- **Giai đoạn 5:** Agent đánh giá cho bảo đảm chất lượng và cải tiến liên tục

Điểm cốt lõi: **kiến trúc của bạn nên tiến hóa theo nhu cầu.** Bắt đầu đơn giản, đo mọi thứ, chỉ thêm độ phức tạp khi nó mang lại giá trị đo được. Kiến trúc tốt nhất là kiến trúc đơn giản nhất đáp ứng yêu cầu hôm nay trong khi mở đường cho năng lực ngày mai.

Chiến lược kiến trúc lai

Khung quyết định cho điểm khởi đầu rõ ràng, nhưng các hệ thống production thường tiến hóa thành kiến trúc lai kết hợp nhiều mẫu một cách chiến lược. Hiểu các tổ hợp này giúp tránh ngõ cụt kiến trúc và mở đường mở rộng có hệ thống.

Các mẫu lai phổ biến:

Hệ phân cấp với xử lý song song

- Agent giám sát giao việc cho agent chuyên trách; các agent chuyên trách điều phối quy trình song song. Ví dụ, giám sát đánh giá rủi ro tài chính có thể giao cho agent rủi ro tín dụng, thị trường và vận hành, mỗi agent chạy các phân tích song song trong lĩnh vực của mình.

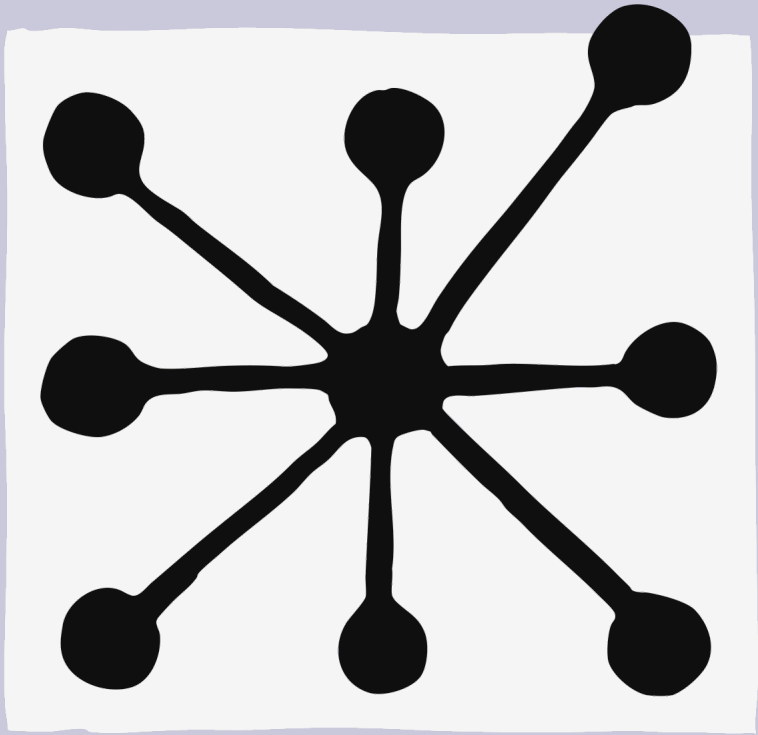
Quy trình tuần tự với định tuyến động

- Tiến trình tuyến tính gọi các loại agent khác nhau theo kết quả trung gian. Quy trình chăm sóc khách hàng có thể bắt đầu bằng phân loại, rồi tùy độ phức tạp của vấn đề mà chuyển sang agent xử lý đơn giản hay một đội nghiên cứu đa agent phức tạp.

Agent đơn với leo thang đa agent

- Agent đơn giản xử lý việc thường nhật nhưng tự động kích hoạt hệ đa agent tinh vi khi gặp trường hợp biên. Cách này tối ưu chi phí trong khi vẫn giữ năng lực cho kịch bản phức tạp.

Hãy nhớ, bạn không bị trói vào các mẫu kiến trúc đơn giản. Khi nhu cầu kinh doanh xứng đáng với độ phức tạp tăng thêm, kết hợp các mẫu một cách chiến lược có thể mở khóa những năng lực mà cách tiếp cận đơn lẻ không đạt được.



Chương 4

Nhìn về phía trước: tương lai của việc xây dựng AI agent

Chương 4

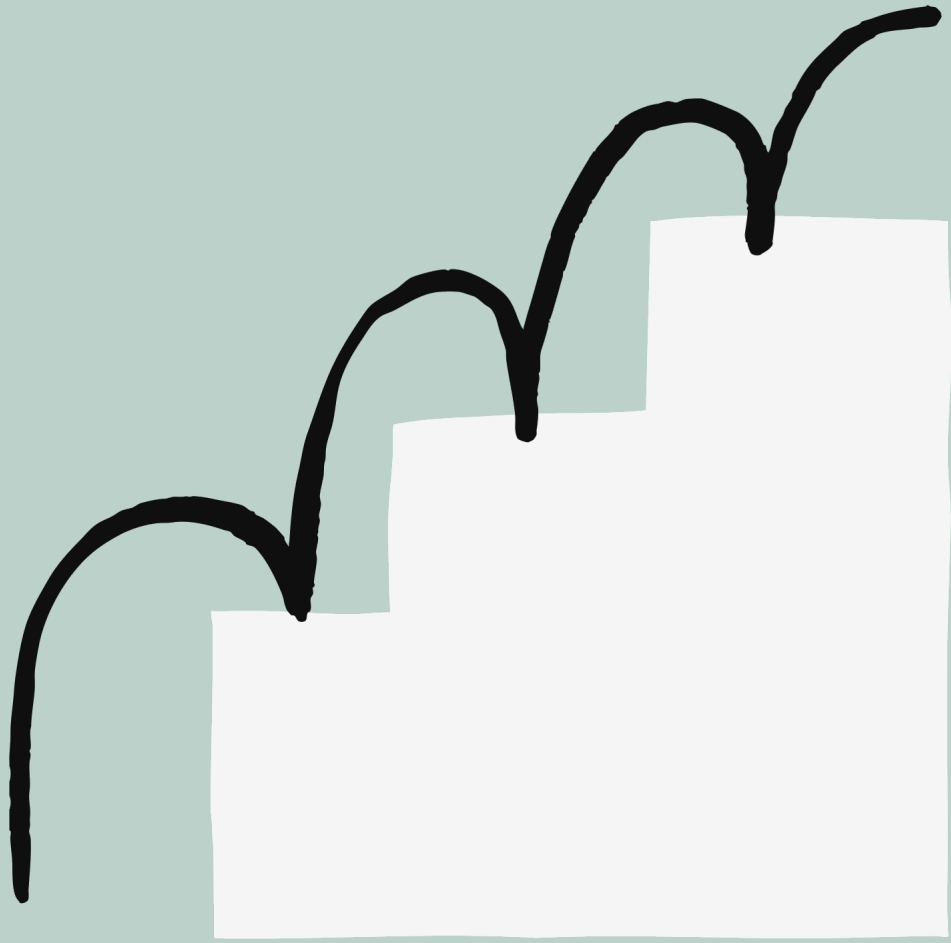
Nhìn về phía trước: tương lai của việc xây dựng AI agent

Trong cẩm nang này, chúng ta đã đi qua trọn phổ triển khai AI agent, từ hệ agent đơn xử lý nhiệm vụ tập trung đến kiến trúc đa agent tinh vi giải các thách thức phức tạp, đa lĩnh vực. Chúng ta đã xem các tình huống thực tế ở nhiều ngành, khám phá các mẫu kiến trúc cùng những đánh đổi của chúng, và sau cùng dựng các khung để ra quyết định triển khai có cơ sở. Với nền tảng này, bạn có thể tự tin xây những AI agent giải quyết vấn đề thật và mang lại kết quả thật.

Triển khai AI agent thành công đòi hỏi cân chỉnh độ phức tạp kỹ thuật với giá trị kinh doanh, thay vì chạy theo kiến trúc tinh vi nhất có thể xây. Bạn sẽ thấy kết quả tốt nhất nếu bắt đầu với agent đơn để chứng minh ROI, xây hệ thống quan sát được từ ngày đầu, và tiến hóa kiến trúc theo những gì dữ liệu nói với bạn. Các tổ chức đi theo cách tiếp cận chừng mực này luôn vượt những ai thiết kế thừa ngay từ vạch xuất phát. Các khung và mẫu chúng ta đã bàn cho bạn nền móng vững, nhưng cách triển khai cụ thể sẽ tùy khẩu vị rủi ro, giới hạn nguồn lực và mức độ sẵn sàng của tổ chức bạn với hệ thống tự hành.

Tổ chức có thể xoay nhanh giữa cách tiếp cận đơn giản và phức tạp khi yêu cầu kinh doanh tiến hóa chính là tổ chức chiến thắng. Dù bạn triển khai một agent chăm sóc khách hàng hay điều phối các hệ nghiên cứu đa agent, Sao Bắc Đẩu của bạn phải là thiết kế module, observability toàn diện và các chỉ số thành công rõ ràng gắn thẳng với kết quả kinh doanh.

Công cụ đã sẵn, cẩm nang đã viết xong. Giờ là lúc giải những bài toán thực.



Chương 5

Bước tiếp theo

Chương 5

Bước tiếp theo

Sẵn sàng xây chưa? Hãy bắt đầu với Claude Developer Platform để tiếp cận các mô hình, công cụ và tài liệu kỹ thuật bạn cần cho việc triển khai AI agent. Dù bạn đang thử nghiệm hệ agent đơn đầu tiên hay mở rộng lên kiến trúc đa agent, những tài nguyên này sẽ tăng tốc quá trình phát triển:

- **Bắt đầu xây agent với Claude Developer Platform** - Truy cập tài liệu API đầy đủ, hướng dẫn triển khai và kỹ thuật viết prompt.
- **Khám phá Agent Skills** - Học cách trang bị cho agent kiến thức chuyên môn, quy trình và tích hợp tool.
- **Xem Building the future of agents with Claude** - Đào sâu các kiến trúc nâng cao và mẫu mới nổi cùng những người đứng sau Claude Developer Platform.
- Khám phá Anthropic Engineering Blog - Bài viết kỹ thuật về **phát triển agent**, **context engineering** và **chiến lược triển khai production**.

Liên hệ **đội Sales của chúng tôi** để tìm hiểu thêm, hoặc đăng ký **Claude Developer Platform** ngay hôm nay.

Bản tiếng Việt được biên dịch và việt hóa bởi Phong Hồ.

Theo dõi thêm bài viết & tài nguyên hữu ích về AI → **phongminhho.substack.com**

Claude

https://www.claude.com/platform/api