May 17, 2026

AskBeevs - How to build agents that do work

Summary

The meeting explored AI agent frameworks and deployment strategies for automating complex workflows via professional skill integration.

Defining AI Agent Frameworks
AI agents function as autonomous systems using frameworks like Claude Code to solve specific, tangible goals. Users can interact via simple interfaces or full-stack applications.

Developing Practical Audit Tools
Agents demonstrate high value in automating compliance audits by analyzing data sets against specific policy constraints. Development requires careful API configuration and environmental setup for successful execution.

Workforce Evolution and Strategy
The primary decision is to transition staff toward leveraging agents for manual tasks to bridge the performance gap between organizations. Success requires domain knowledge paired with technical communication skills.

Rate this Summary: Helpful or Not Helpful

Next steps

- ☐ [Timothy Beevor] Present Design Tool: Demonstrate the design tool functionality in the upcoming session. Explain how to utilize this new feature effectively.
- ☐ [Sanjana Gupta] Watch Anthropic Session: Watch the 20 minute session provided by Anthropic on how to build agents.
- ☐ [Sanjana Gupta] Create Rental Agent: Attempt to create a personal agent for summarizing house rentals in Jalong by configuring email notifications and using Claude to parse the data.
- ☐ [Sanjana Gupta] Discuss System Integration: Schedule a follow-up discussion with Timothy Beevor to address the integration challenges between payment, onboarding, and timesheet systems in the office.

Details

- **Introduction to AI Agents**: Timothy Beevor defines AI agents as autonomous systems that, when provided with a tangible output goal, work to solve a specific problem. Agents operate across various modalities, including compliance auditing, content creation, ordered workflows, knowledge retrieval, conversational engines, and event-driven monitoring. These frameworks are accessible to non-technical users because AI tools can assist in building them end-to-end.
- **Agent Frameworks and Interfaces**: Agents function within specific frameworks such as LangGraph, LangChain, Crew AI, and Claude Code, which act like programming languages for the agent. Users can interact with these agents through various interfaces, beginning with a Command Line Interface, evolving to simple user interfaces like Streamlit, or advancing to full-stack applications where agents can programmatically modify the interface itself.
- **Technical Considerations and Tool Use**: When building an agent, users must select an AI model—often accessed via API—and consider data storage requirements. Agents can utilize tools to perform tasks on behalf of the user, such as accessing data via a retrieval-augmented generation model or interacting with other applications through the Model Context Protocol.
- **Deployment and Agent Lifecycle**: Agents can run locally on a personal computer or be deployed to cloud environments. Once an agent is developed,

it can assist in its own deployment, including building databases, managing code on GitHub, and fixing errors during the deployment process.

- **Professional Innovation and Security**: Timothy Beevor cautions against using confidential work information in agents. Instead, they suggest that organizations create "safe spaces" where employees can use synthetic data to innovate, build, and test software without violating security policies.
- **Integrating Agents into Applications**: Agent functionality can be embedded into existing software applications to enable dynamic workflows. For example, an application can use an agent to make decisions or generate UI components in real-time based on user input, though this approach may be expensive at scale due to the high volume of API calls.
- **Project Scenario - Credit Card Policy Audit**: To demonstrate building an agent, Timothy Beevor introduces a scenario involving monitoring credit card transactions against the Melton City Council policy. The process involves finding policy documents, understanding finance system data structures, and generating a synthetic dataset of 300 transactions to test compliance, gray-area, and fraudulent activity.
- **Tool Selection - Claude Code versus Claude Co-work**: When developing the agent, Timothy Beevor explains that Claude Co-work is better suited for knowledge work, such as writing reports, while Claude Code is designed for "hardcore" software development that continues until the task is complete.
- **Developing the Audit Agent**: Timothy Beevor demonstrates the prompt-writing process within Claude Code, instructing the agent to audit transactions in the provided CSV file against the Melton City Council policy. The agent is directed to create a self-contained, user-friendly script that allows a beginner to run the audit from a Command Line Interface.
- **Agent Computation and Output**: Users can adjust the "level of thought" in the agent, with higher levels potentially reducing errors at the cost of more compute usage. Upon execution, the agent generates essential files, including a Python script, a requirements file, and a readme document containing instructions for the user.
- **Environment and API Setup**: After the agent generates the code, the user must configure the environment by pasting an Anthropic API key into the designated environment file. Once saved, the user can execute the audit script via the terminal.
- **Execution and Result Analysis**: Running the agent involves executing the Python script, which processes transactions in batches. The agent produces

an audit report that categorizes transactions by compliance tier and provides rationale for its findings based on the policy clauses.

- **Developing a User Interface with Streamlit**: To improve accessibility, Timothy Beevor demonstrates how to instruct the agent to build a Streamlit-based user interface. This allows the end user to input transaction metadata and upload CSV files through a web interface rather than using the command line.
- **Advanced Multi-Agent Workflows**: Systems can be designed to include multiple agents, each with specialized tasks. Timothy Beevor cites an example of an assignment-review system where 12 agents perform distinct jobs—such as checking grammar, coherence, and research accuracy—before a final agent synthesizes the feedback.
- **Advanced Reporting and Document Generation**: The agent can be updated to include advanced features, such as generating professional Microsoft Word reports in specific templates for an audit committee. This functionality demonstrates how agents can be tasked with formatting output to meet specific professional standards.
- **Debugging and System Integration**: If errors occur during development or execution, users can copy and paste console error messages back to the agent, which can then identify and fix the issue. Agents can also be used to connect different modules, effectively building and appending new features to the existing user interface.
- **Future of Coding and Professional Impact**: Timothy Beevor argues that while traditional coding requires specialized departments, the ability to write code is essential for high-performing AI users. They emphasize that workers should be trained to use agents to handle repetitive, boring tasks, potentially using burner machines to bypass organizational security bottlenecks that slow innovation.
- **Real-World Example - Engineering Design Comparison**: Randal Tait shares a practical application where they built an agent to compare an engineering design model against a structural design model, verifying components beam-by-beam and column-by-column to produce a final compliance report.
- **Workforce Evolution**: Timothy Beevor concludes that rather than fearing job automation, organizations should train staff to leverage agents for manual, time-consuming tasks. This shift allows workers to focus on higher-level compliance measures and strategic follow-up actions once the agents have processed the bulk data.
- **AI and Document Formatting**: Timothy Beevor argues that current AI models can perform better than human formatting in traditional word processors by

writing JavaScript to generate well-structured documents. Timothy Beevor notes that while older AI struggled with proprietary document formats, modern AI can interpret open, interoperable standards effectively, making many traditional software features unnecessary.

- **Deploying AI Agents for Personal and Business Use**: Timothy Beevor discusses the feasibility of deploying AI agents to the cloud with password protection for personal or small business use cases that do not involve sensitive financial or customer data. Timothy Beevor identifies specific deployment tools, such as Streamlit and Railway with Docker, and suggests that developers can achieve a full-stack, server-based deployment in approximately 45 minutes.
- **Version Control and GitHub Integration**: Randal Tait and Timothy Beevor discuss the importance of version control when connecting AI to local folders to prevent unintended data overwrites. Timothy Beevor explains that even for those not skilled with GitHub, AI can assist in managing repositories and backing up work to ensure data safety.
- **Real Estate Rental Search Challenges**: Sanjana Gupta proposes building an agent to summarize rental listings based on specific criteria, but Timothy Beevor warns that such projects face significant hurdles. Timothy Beevor explains that major rental platforms actively defend against bot activity, and attempting to scrape their data directly can be expensive due to the high cost of the necessary "computer use" licensing.
- **Enterprise System Integration and Audit Automation**: Sanjana Gupta describes a challenge regarding the lack of integration between enterprise systems for payments, onboarding, and time-sheeting, which leads to reconciliation errors and compliance risks. Timothy Beevor suggests a solution where a local agent extracts non-sensitive transaction data from these systems on a daily basis to flag compliance issues, creating an audit trail that documents inputs, outputs, and timestamps. Timothy Beevor further suggests that tools like Google Command Line Interface or Google Antigravity could be used for this automation, pending approval from an IT department.
- **Refined Strategy for Rental Searches**: Randal Tait and Timothy Beevor develop a two-stage model for the rental search problem, shifting away from direct scraping to avoid blocking. Timothy Beevor advises creating a dedicated Gmail account with scoped OAuth access, setting up automated email notifications from real estate websites, and using an AI agent to parse those incoming emails against specific search criteria to generate a report.
- **Future Deployment Engineers and Essential Skill Sets**: Sanjana Gupta introduces the concept of "Future Deployment Engineers," noting that the role

requires a synthesis of three core skills: domain knowledge, technical proficiency, and communication or strategy skills. Timothy Beevor agrees, emphasizing that while domain expertise is valuable, it can often be sourced from others, whereas the ability to effectively utilize AI, combined with persuasive communication to navigate corporate governance, is essential for success.

- **Corporate AI Adoption and Performance Impacts**: Randal Tait reports that companies providing staff with unlimited Anthropic tokens are currently hiring, while those that have banned AI tools like Copilot are experiencing layoffs. Timothy Beevor notes that this cultural difference will likely lead to an exponential performance gap between organizations that embrace AI and those that restrict it.
- **Critique of Microsoft Copilot**: During the discussion on corporate AI tools, Sanjana Gupta, Bruce, and Timothy Beevor critique Microsoft Copilot, with Timothy Beevor characterizing the tool as a poor-performing product compared to other available AI options.
- **Custom Software Development via AI**: Randal Tait shares a success story where they used AI to build a bespoke project management tool in HTML, allowing their family to track renovation progress without needing to purchase or learn external software. Randal Tait emphasizes that this "job done" approach allows individuals to create specific solutions tailored to their needs without relying on traditional software providers.
- **Leveraging Open Source Software**: Timothy Beevor concludes by suggesting that users can find existing open-source software that aligns with their goals and utilize AI to edit, fork, or customize that code. Timothy Beevor notes that because AI models are trained on this public software, they are well-equipped to help users build complex modules such as geographic information systems, computer-aided design tools, or finance applications.

You should review Gemini's notes to make sure they're accurate. Get tips and learn how Gemini takes notes

*How is the quality of **these specific notes?** Take a short survey to let us know your feedback, including how helpful the notes were for your needs.*