

WHITEPAPER

Building Generative AI-Powered Applications on **Google Cloud** From Architecture to Production

Table of **CONTENTS**

- 01** What Makes an Application GenAI-Powered?
- 02** Role of Agentic AI
- 03** Building GenAI and Agent-Powered Applications with Gemini Enterprise Agent Platform
- 04** Grounding GenAI and Agents with Enterprise Knowledge
- 05** GenAI in Practice
- 06** Engineering for Context, Quality, Cost, and Trust
- 07** Deployment and GenAI/Agent Lifecycle Management
- 08** Building an Enterprise GenAI Application Strategy on Google Cloud

This paper aims to give readers a clear, practical overview of Google Cloud's Generative AI offerings and how enterprises can leverage them to build production-ready, GenAI-enabled applications. It covers the core building blocks: foundation models, Agentic-AI, Retrieval-Augmented Generation (RAG), embeddings, and vector search and explains how these components come together to support real-world applications.

Beyond the platform capabilities, this paper highlights Cybage's role in helping customers navigate this landscape: from selecting the right models and architecture patterns to designing scalable RAG pipelines and integrating LLMs into existing enterprise systems. The goal is to equip technical and business stakeholders with a practical understanding of what's possible today, and a framework for evaluating how GenAI can be applied within their own organization.

What Makes an Application GenAI-Powered?

Enterprise applications are usually based on fixed logic. Users provide inputs, and applications process them using programmed rules and workflows to produce expected outputs. Although this method remains fundamental to enterprise software, it becomes limiting when applications must understand intent, handle unstructured information, or respond dynamically to changing context.

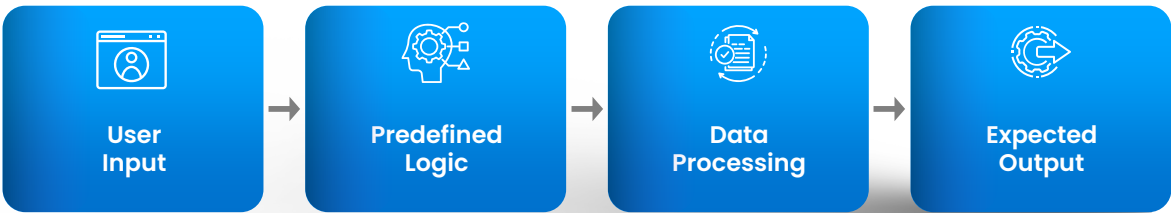
Generative AI (GenAI) introduces a new approach to application development. GenAI-based applications can interpret natural language, understand context, produce content, and draw on available knowledge to provide appropriate responses.

A GenAI-powered application integrates foundation models into its user experience, workflows, or application logic. This enables conversational interaction, summarization, content generation, knowledge discovery, and contextual assistance.

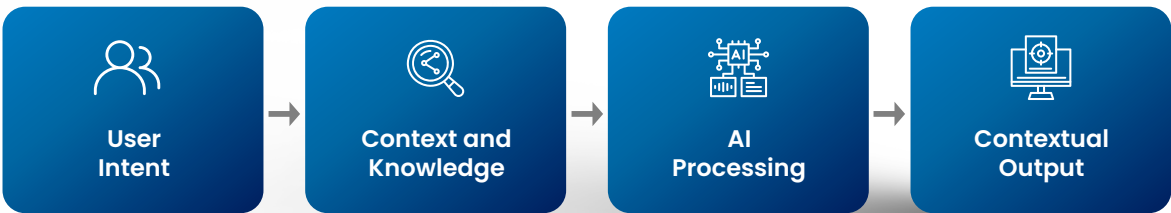
However, integrating a Large Language Model (LLM) alone does not create a production-ready GenAI application. Model intelligence must work with business context, trusted enterprise information, existing applications, and appropriate controls.

The application pattern consequently shifts from:

Traditional



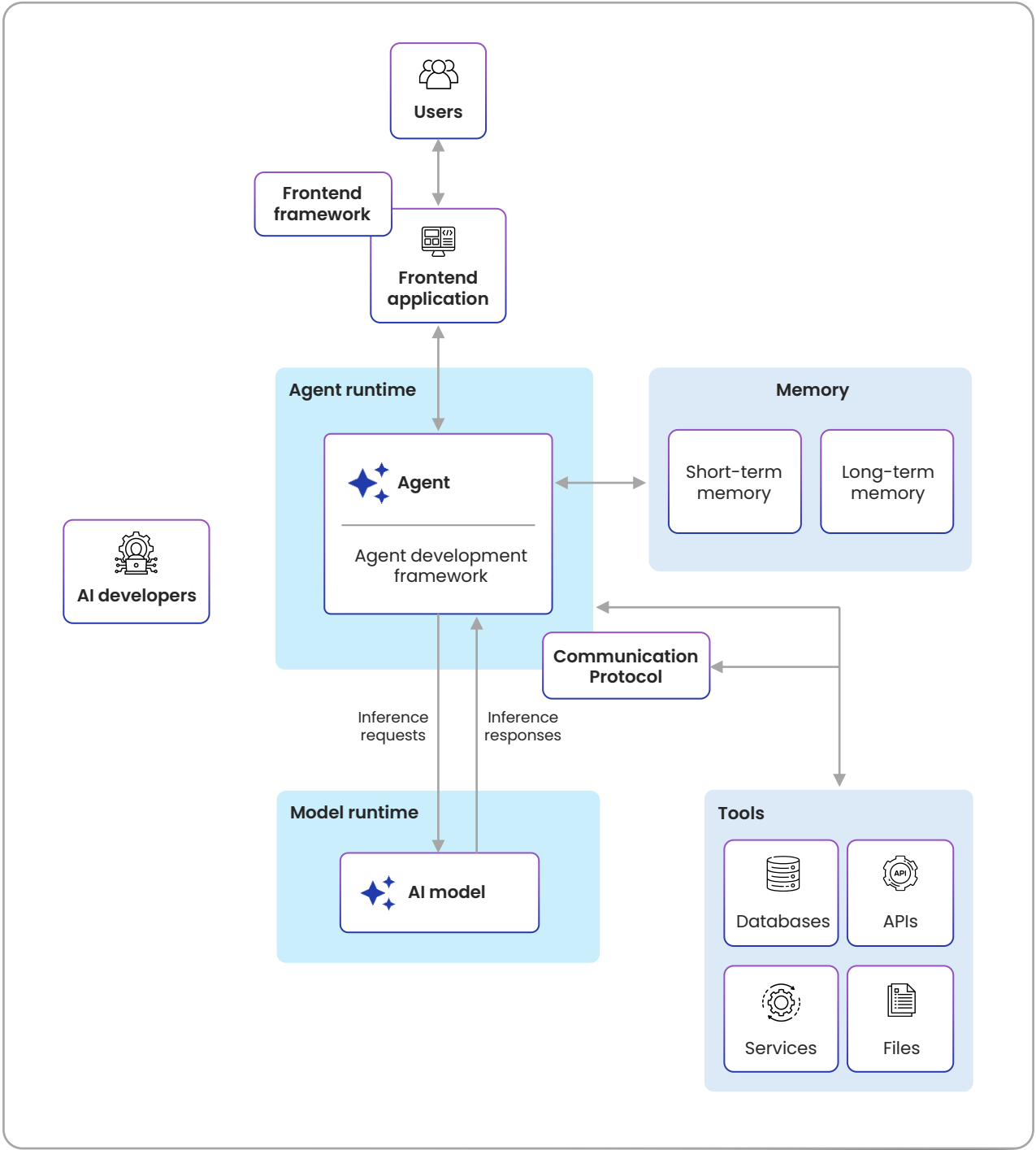
GenAI-powered



Instead of relying on strict navigation, search queries, or predefined workflows, users can state their requirements in natural language. This approach enables more intuitive, personalized, and knowledge-aware interactions for end users, while for application owners, GenAI can speed up feature innovation, automate knowledge-intensive tasks, and provide capabilities that would otherwise require extensive rules and application logic.

Role of Agentic AI

Agentic AI is an advanced form of GenAI application Architecture that is focused on autonomous decision-making and action. Unlike traditional AI, which primarily responds to commands or analyzes data, agentic AI can set goals, plan, and execute tasks with minimal human intervention.

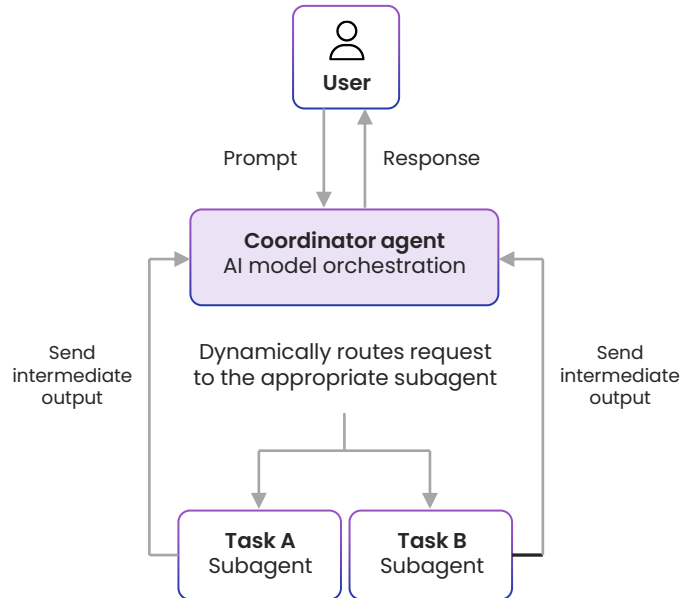


An AI agent is a software component that achieves a goal by processing input, performing reasoning with available tools, and taking actions based on its decisions. AI agents use function calling to format the input and ensure precise interactions with external tools. Agent design patterns are common architectural approaches to build agentic applications. It offers a distinct framework for organizing a system's components, integrating the model, and orchestrating a single agent or multiple agents to accomplish a workflow. A few of the patterns are illustrated below which can be seen as a guideline to build agentic patterns while final solution can be custom tailored based on the need.

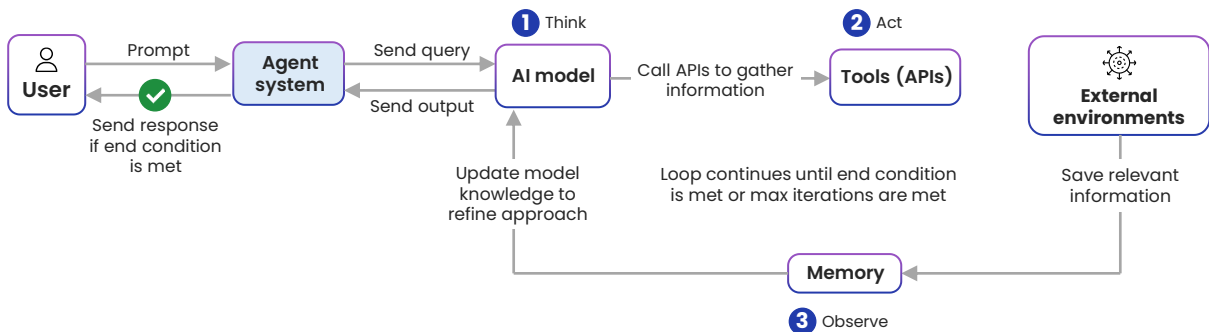
A high-level view of a single agent pattern



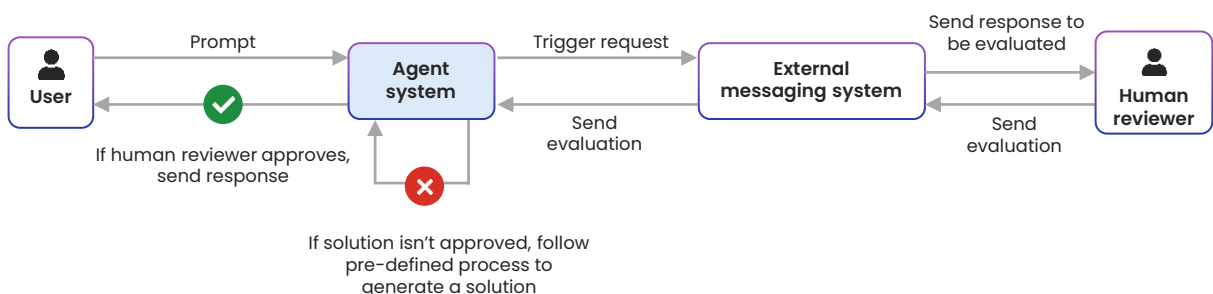
A high-level view of a multi-agent coordinator pattern



A high-level view of the ReAct pattern



A high-level view of a human-in-the-loop pattern



Building GenAI and Agent-Powered Applications with Gemini Enterprise Agent Platform

As enterprises progress toward agent-powered applications, the underlying platform must support more than model access. Models, enterprise knowledge, tools, orchestration, applications, and operational controls need to work within a connected environment.

Gemini Enterprise offers a basis for combining AI agents, enterprise data, applications, and users. As part of the wider Google Cloud AI ecosystem, Gemini models, Vertex AI features, and Google's Agent Development Kit (ADK) support the development and deployment of generative AI and agent-driven applications.

Gemini Enterprise Agent Platform is a unified platform to build, deploy, govern, and optimize enterprise-grade AI agents and model-based solutions. As an evolution of Vertex AI, it supports the complete AI lifecycle from accessing foundation models to deploying and managing your agents. It provides low code development model via Agent studio as well as code-based development environment. Although the platform provides complete management of AI agents, for custom needs we can still do development using various agentic frameworks and deploy such agents in service of choice like Cloud-Run or Kubernetes.

Selecting Gemini Models

GenAI experiences on Google Cloud use Gemini models. These models support different data types, so your applications can work with text, documents, images, audio, video, and code. When picking a model, think about your specific tasks. Complex tasks need more reasoning ability, while tasks that process a lot of data should focus on speed and efficiency. Choose a model that fits your needs for context, throughput, latency, and cost, rather than always picking the most advanced option.

Generative Models

		
3.8 Flash Best for tackling complex agentic tasks at scale	3.5 Flash-Lite Best for high-volume tasks that need efficiency and intelligence	3.1 Pro Best for complex tasks and bringing creative concepts to life

Prompts, Tokens, and Context

Prompts set instructions, limitations, and expected behavior. Tokens represent the information the model processes and produces, and the context window specifies how much information the model can consider during an interaction. Enterprise context may include system instructions, conversation history, retrieved information, documents, and application data. Prompt engineering therefore goes beyond simply writing instructions, since it involves intentionally arranging the information available to the model.

Gemini's multimodal capabilities also extend here, as applications can interpret various types of content within one experience, enabling document processing, visual understanding, content generation, and other multimodal workflows.

Agentic Framework and Runtime

Based on this progress, Google has developed an Agent Development Kit (ADK) which offers a framework for creating and coordinating agents. Developers can specify agent behavior, link tools, and set up workflows that involve individual or specialized agents.

The essential components include reasoning to determine how to carry out a task, using functions to call external capabilities, using memory to retain relevant information, and applying multi-agent patterns to coordinate specialized tasks. Through APIs and tools, the agent can interact with databases, enterprise applications, and business services while those systems continue their controlled business functions.

Production agents may also need to maintain state or run workflows that extend beyond a single interaction. Managed agent runtime capabilities provide an execution foundation for deploying and scaling these workloads.

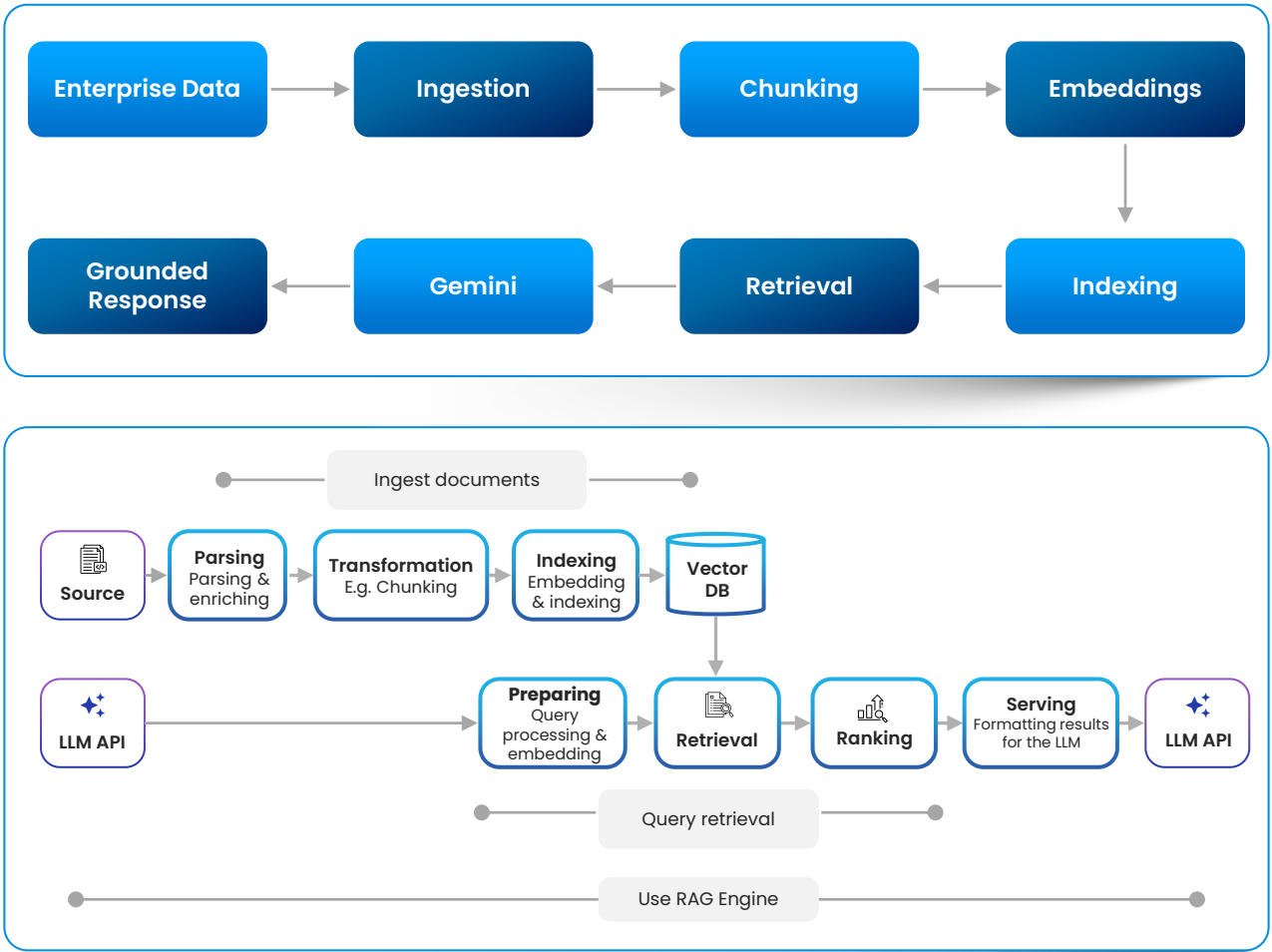
Grounding GenAI and Agents with Enterprise Knowledge

Although foundation models are powerful, many enterprise applications depend on proprietary, domain-specific, or constantly changing information that they cannot rely on solely from pretraining. Grounding links models to the right organizational information at the time of interaction.

Building RAG on Google Cloud

RAG enables applications to retrieve relevant enterprise information and provide it to the model as contextual evidence.

A typical flow is



Source: [Agentic AI architecture guides](#) | [Cloud Architecture Center](#) | [Google Cloud Documentation](#)

The system takes in documents and breaks them into clear sections. It then uses embeddings to turn their meaning into numbers, which are stored in an index for later use. When someone makes a request, the application looks up the most relevant information and gives it to Gemini as context. Vector Search makes this process better by finding information based on meaning, not just matching keywords. This helps users find what they need, even if they use different words than those in the company's documents.

Preparing Enterprise Knowledge

The quality of RAG is very much dependent on the way in which you prepare knowledge; when chunking it is necessary to retain sufficient meaning without including any unnecessary information and metadata can be used to improve retrieval by means of attributes such as document type, geography, date, business function, or access level.

Enterprise AI should also handle various data formats. Semantic retrieval works well for documents, manuals, images, presentations, and knowledge repositories, while frequently changing structured information such as transaction status or account information is better obtained directly via authorized APIs or tools. It is just as important to consider freshness and permissions. Indexes must keep up with updates to the source material, and retrieval must account for user identity, data classification, and application access controls.

For agents, this knowledge becomes more than material for answering questions. Retrieved enterprise knowledge can inform both agent reasoning and subsequent actions.



GenAI in Practice

Most GenAI applications don't start from scratch; instead, they build on data platforms, cloud-native applications, integrations, analytics environments, and operational foundations already in place to support enterprise workloads.

Cybage has designed and implemented Generative AI and RAG-based solutions on Google Cloud, helping customers move from raw data and legacy pipelines to intelligent, agent-driven applications. These engagements typically involve identifying high-value GenAI use cases, planning data accessibility and pipeline architecture, building custom tools for agent-tool calling, and implementing vector search for grounded, context-aware responses.

The following cases illustrate such implementations.

1 | Intelligent Health Monitoring & Reporting System

A large-scale digital enterprise needed to continuously scan and assess the health of thousands of web pages and sites: spanning performance, availability, content quality, and other operational metrics. The existing process relied on raw dashboards and manual analysis, making it difficult for stakeholders to quickly interpret results or get actionable, plain-language insights at scale.

GenAI-Enabled Solution

The customer built a GenAI-powered reporting layer on top of their existing scanning infrastructure, using Gemini models to translate raw site health data into intuitive, natural-language insights and reports. Rather than requiring users to interpret dashboards manually, the system allows users to query site health conversationally and receive synthesized, contextual answers.

Architecture Highlights

Multi-Agent Design (built with ADK)

The solution follows a multi-agent pattern rather than a single monolithic agent. A main orchestrator agent handles request routing and final summarization, while specialized sub-agents each own a specific domain area, allowing focused reasoning and cleaner tool boundaries per agent.

Tool-Calling Layer

Custom tools were built to expose underlying site metrics data to the agents in a structured way, enabling agents to call the right tool for the right data rather than relying on static context — a core RAG/agentic pattern for grounding responses in real, current data.

Vector Search with Hybrid Retrieval (AlloyDB)

Content and report data are indexed in AlloyDB, using a hybrid search approach that combines BM25 (keyword/lexical search) with pgvector (semantic/embedding-based search). This hybrid approach improves retrieval accuracy — capturing both exact metric/term matches and semantically related content that a single search method might miss.

This pattern demonstrates how GenAI and RAG can be layered onto existing large-scale data pipelines without needing to re-architect the underlying scanning system — turning raw operational data into accessible insights via a conversational, agentic interface.

2 | AI-Powered Executive Event Companion App

A global enterprise running executive flagship events faced fragmented event management across multiple platforms. Attendees had disconnected access to agendas, notifications, and networking tools, which limited insight capture, social content generation, and the premium experience expected at executive-level events. Additional challenges included maintaining low latency and real-time session sync for hundreds of concurrent executive users, alongside tight timelines requiring rapid turnarounds across multiple events with custom variations.

GenAI-Enabled Solution

The team built a unified, web-based companion app centralizing event logistics, agendas, and real-time alerts into a single experience. Key capabilities included:

Built-in, dedicated professional networking features to facilitate meaningful connections between attendees.

Structured gamification and audience incentives to drive active exploration and participation throughout the event.

An all-in-one suite for capturing insights, accessing official event media, and generating AI-powered social content — including customized LinkedIn sharing integration. Leveraged AI-assisted workflows to rapidly build, test, and iterate tailored variations across events.

Engineering for Context, Quality, Cost, and Trust

Moving GenAI into production introduces engineering considerations that conventional application development does not address in the same way. Teams must control how much information a model processes, how reliably it responds, how quickly interactions complete, and what they cost.

Optimizing Context and Cost

Adding more context does not always improve results. Pruning removes outdated information, while summarizing and compacting preserve key points concisely. Caching prevents unnecessary reprocessing of unchanged content.

The same principle should also apply to RAG retrieval. Metadata filters, ranking, chunk size, and retrieval limits help surface the most relevant evidence without unnecessarily increasing the context sent to the model. This achieves a practical balance between response quality, time delay, and cost. For tasks that require a high degree of reasoning, it may be worthwhile to use more capable models and provide additional context, while in situations involving a high volume of interactions, targeted retrieval and more efficient models may be advantageous.

We can control the model's reasoning capabilities by adjusting the [thinking budget](#), or the number of thinking tokens, based on task complexity. A higher thinking budget lets the model perform more detailed reasoning and planning before it provides an answer. A higher thinking budget can improve response quality, but it might also increase latency and cost.

To optimize performance and cost, implement [model routing](#) to dynamically select the most appropriate model for each task based on the task's complexity, cost, or latency requirements. For example, you can route simple requests to a small language model (SLM) for structured tasks like code generation or text classification and reserve a more powerful and expensive model for complex reasoning. If we implement model routing in your agentic application, we can create a cost-effective system that maintains high performance.

Engineering for Reliable AI

Responses from GenAI applications may appear credible but can be incomplete or lack support, so production systems must have safeguards beyond the model itself.

Applications can restrict the model's instructions, require output to stay within the provided evidence, check responses against business rules or defined schemas, and refuse to answer when supporting information is inadequate. Quality thresholds should reflect business risk. A creative assistant can tolerate greater variation than an agent performing consequential enterprise actions.

Security should also extend across the complete AI interaction. Authentication and authorization should govern access to models, information, and tools. Control sensitive data before it enters model context and restrict agent permissions to the actions it can execute.

Responsible AI practices should address inappropriate outputs, bias, explainability, and the consequences of automated decisions based on the application's risk profile.

Measuring Quality and Trust

GenAI evaluation should account for outputs that can vary yet still be acceptable. Teams may use appropriate evaluation datasets to assess aspects such as relevance, groundedness, adherence to instructions, and task completion.

Traceability helps explain how an agent reached a specific result by showing what information it used and which tools it interacted with. For higher-risk tasks, human oversight may be needed to approve important actions or to review uncertain results.

Deployment and GenAI/Agent Lifecycle Management

Production GenAI applications introduce a broader set of software lifecycle requirements. Models, prompts, retrieval configurations, tools, and agent definitions can change independently and therefore need to be managed as versioned and testable components.

From Experimentation to Production

CI/CD practices can extend traditional application delivery to include AI-specific assets, and development teams can validate the application code alongside prompts, agent definitions, tool integrations, and configuration before promoting a release. A change to the prompt or model can affect application behavior even when the underlying code remains unchanged. Clearly defined versions improve traceability, enable rollbacks, and make it easier to use developing Gemini features without tightly linking the application to a specific model implementation.

Automated release validation should include deterministic testing of APIs, integrations, permissions, and business logic, along with AI-specific assessment. Moreover, the agent workflows should be tested regarding tool selection, task completion, failure handling, and authorization boundaries.

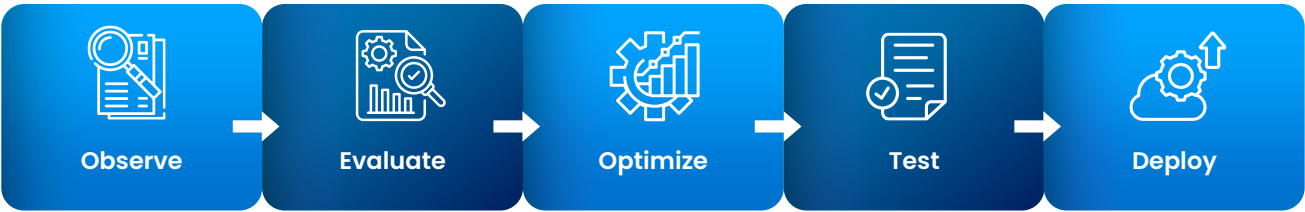
Operating Agents at Scale

Agentic workloads can involve persistent state, multiple tool calls, and longer runtimes. The managed runtime separates the agent's logic from the infrastructure required to operate and scale it.

Once teams deploy agents, they should have visibility into latency, errors, token consumption, model usage, tool failures, and cost. Agent execution should also be attributable, so organizations know which agent acted, what resources it accessed, and whether the workflow completed successfully.

Production telemetry then serves as the basis for improvements. Teams can refine the models, prompts, retrieval settings, or agent logic using observed behavior and user feedback before going through the evaluation and deployment process again.

This creates a continuous lifecycle:



Building an Enterprise GenAI Application Strategy on Google Cloud

Enterprise GenAI is moving beyond isolated features and experiments toward reusable application capabilities. The strategic opportunity is to establish a modular foundation where models, enterprise knowledge, agents, and existing applications can evolve independently while remaining connected through consistent engineering patterns. This changes the development mindset. Rather than creating separate knowledge of pipelines and integrations for every use case, organizations can establish common grounding patterns and reusable approaches for connecting AI with enterprise systems.

As the capabilities develop, application architectures will gradually shift from being model-centric to agent-centric. Although models will still form the intelligence layer, enterprise value will increasingly come from how effectively agents use trusted knowledge, tools, applications, and human supervision to carry out significant tasks. Google Cloud is providing the components needed for this evolution through Gemini, enterprise grounding, agent development, and managed AI features. The engineering challenge is to integrate these elements into an architecture that supports controlled evolution as technology or business requirements change. For Cybage, this approach is in line with a product engineering method that links emerging AI capabilities with application architecture, data engineering, integration, quality engineering, DevOps, and enterprise systems.



Connect with Cybage to build scalable, production-ready GenAI and agent-powered applications on Google Cloud.

Reach out to us at business@cybage.com to know more.