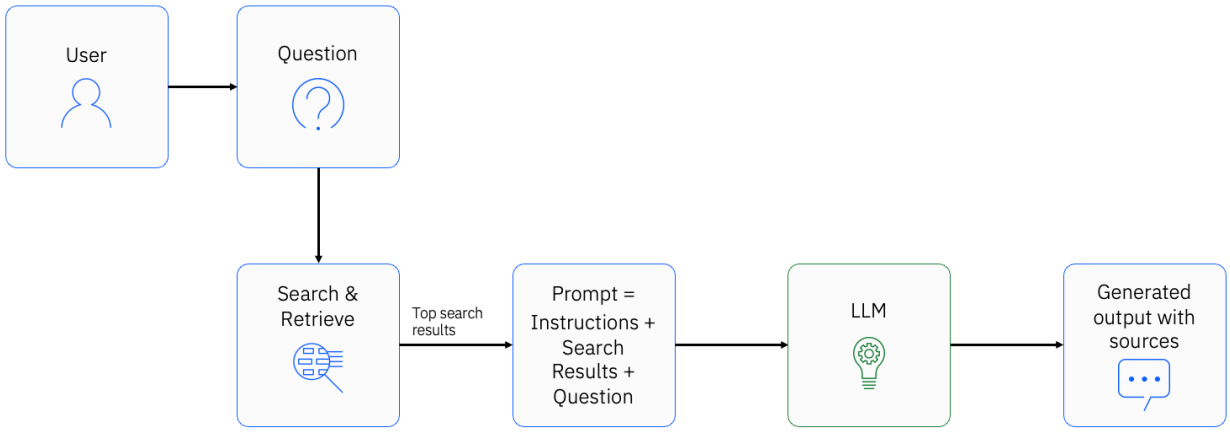
Lab 2: Building RAG agent in Watsonx Orchestrate

Traditional RAG vs Agentic RAG

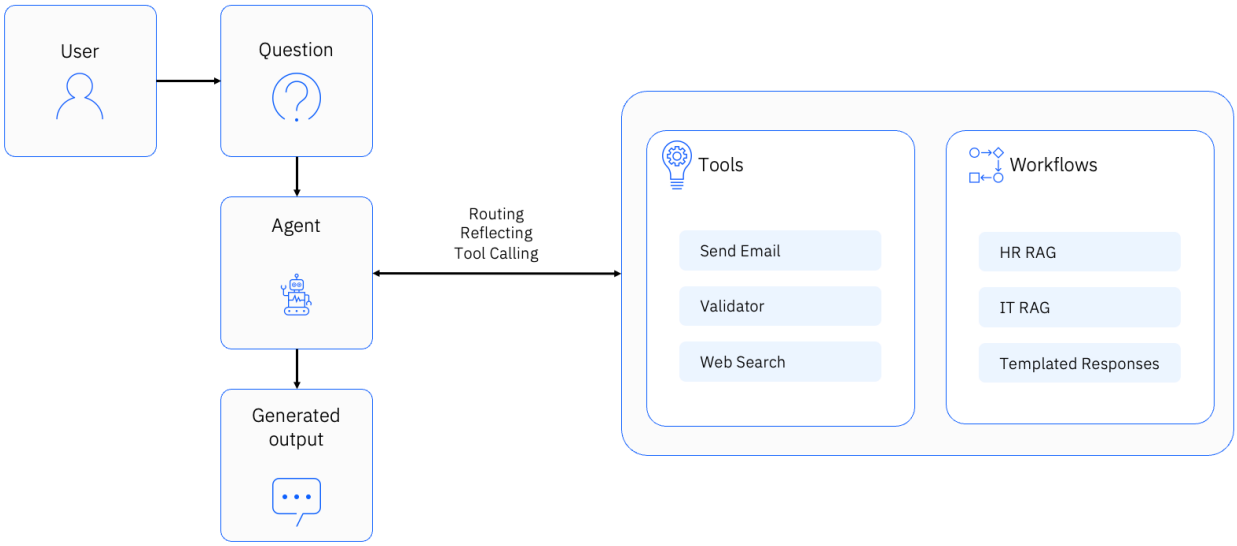


Retrieval-Augmented Generation (RAG) is a widely used method in AI applications where a user submits a question or request to an agent. In a traditional RAG workflow, this input is sent to a search and retrieval system—typically a knowledge base containing internal documents. The system retrieves the most relevant results, which are then combined with instructions and the original question to form a prompt for a large language model (LLM). The LLM generates a response grounded in the retrieved data, which is then returned to the user. This describes the standard process of traditional RAG.

While effective for static use cases, such as FAQ chatbots that answer directly from documentation, traditional RAG has limitations. If a user's question is not covered in the primary knowledge base or exists in a different source, the system follows the same fixed pipeline and may return a generic response like, "I don't have that information." It lacks the ability to make dynamic decisions—such as querying multiple sources, skipping unnecessary retrieval steps, or taking final actions beyond text generation.

For example, consider drafting a resolution email for a customer. Traditional RAG can retrieve the relevant troubleshooting steps, but it cannot perform the next step—sending the email—because its workflow is limited to retrieval and generation.

This illustrates why traditional RAG alone is no longer sufficient for enterprise use cases. Enterprises increasingly require AI that can not only retrieve information but also reason, make decisions, and act on behalf of the user.



This is where **Agentic RAG** extends beyond the limitations of the traditional approach. Rather than following a fixed, single pipeline, Agentic RAG allows the AI to operate as an intelligent agent that can **plan, reason, and take multiple steps**. This enables dynamic decision-making—routing requests to the most suitable agent or tool as needed.

In practice, Agentic RAG can orchestrate multiple specialized agents (such as a Router, Retriever, Grader, or Answer Generator). Each of these agents has a defined role and can validate, refine, or even override outputs from earlier stages. For example, a Grader Agent might evaluate the relevance of retrieved results and, if necessary, trigger a fallback to an external web search.

Beyond retrieval, Agentic RAG can also call external tools to perform actions. For instance, an email-sending tool connected to Outlook can be invoked directly by the agent to complete a user's request—something a traditional RAG pipeline cannot achieve.

This multi-agent orchestration makes the system more **adaptable, reliable, and transparent**, since every decision point is managed by a purpose-built agent rather than a rigid, one-size-fits-all process.

Aspect	Traditional RAG	Agentic RAG
Process Flow	Fixed, single pipeline (retrieve → generate)	Flexible, multi-step workflow (plan → reason → act)
Decision-Making	No dynamic choices; always retrieves from the same knowledge base	Dynamically routes requests to the right agent, tool, or data source
Agents/Components	One pipeline handles all steps	Multiple specialized agents (e.g., Router, Retriever, Grader, Answer Generator)
Adaptability	Limited to the documents in the configured knowledge base	Can adapt, validate outputs, retry with fallbacks (e.g., trigger web search if data is missing)
Actions Beyond Text	Restricted to text generation only	Can invoke external tools and systems (e.g., send an Outlook email, update a record)
Enterprise Value	Suitable for simple, static use cases like FAQs	Enables complex, end-to-end business workflows with higher accuracy and automation