AWS Partner Network (APN) Blog

# How Infosys Built an Enterprise Knowledge Management Assistant Using Generative AI on AWS

by Prantik Gachhayat and Ashutosh Dubey | on 23 OCT 2023 | in Amazon Bedrock, Amazon Kendra, Amazon SageMaker JumpStart, Artificial Intelligence, AWS Partner Network, Generative AI, Intermediate (200), Thought Leadership | Permalink | Comments | Share

*By Prantik Gachhayat, Enterprise Architect – Infosys*
By *Saurabh Shrivastava, Head of Solutions Architecture – AWS*
By *Ashutosh Dubey, Sr. Partner Solutions Architect – AWS*

A common challenge faced by many companies involves the requirement to enhance the clarity and availability of internal documents.

In large organizations, information like project details, application designs, business requirements, support process, and onboarding information are stored as documents in different formats, ranging from Confluence pages and SharePoint documents to Jira tickets and files like Word or PDFs stored in shared drives.

Infosys

Want to work with Infosys?

Connect

Here are some instances where team members find it challenging to access the right information in an effective manner:

- When members from a team try to find information about an application developed by other teams, they have to find the subject matter expert (SME) of that application or search for relevant documents. This is a manual and time-consuming process.
- When a new member joins a team, there are typically lots of onboarding formalities to be completed. To get access to different systems, for example, they may be by sending emails or creating support tickets.
- For teams supporting multiple applications, it's challenging to get the required information or understand the logic of applications required for fixing a production issue. They must go through multiple documents or work with the development team to get the details.

These scenarios present significant hurdles for support teams, business users, and new members who often encounter difficulties locating the relevant documentation. Introducing a form of automation to address this issue can yield substantial benefits. Such automation could enhance document accessibility and content availability, leading to notable time savings and improved productivity.

This post will discuss how **Infosys** built an enterprise knowledge management assistant using generative artificial intelligence (AI) technologies on Amazon Web Services (AWS). Infosys is an AWS Premier Tier Services Partner and

Managed Services Provider (MSP) that enables clients to outperform competition and stay ahead of the innovation curve.

## Generative AI on AWS

Generative AI can create new content and ideas, such as conversations, stories, images, videos, and music. Unlike other AI, it doesn't work with pre-existing data; instead, it uses machine learning (ML) models to generate novel and original content based on the principles of probability and statistics.

One of the key technologies behind generative AI is called a transformer-based neural network architecture. This involves training large models containing billions of parameters or variables, which are then able to manipulate and transform input data in various ways to produce new and unique output. The result is that generative AI can create content that's similar to how humans think and create, making it a powerful tool for a wide range of applications.

Generative AI relies on the intelligence of the foundation models (FM) that are also referred as large language models (LLMs). FMs in generative AI are large-scale neural network architectures that serve as the basis, or foundation, for various generative tasks. These models are pre-trained on vast amounts of diverse data, learning to understand and capture the patterns, structures, and representations present in the data. These models can be fine-tuned for specific tasks or used as a starting point for generating new content in various domains.

Enterprises around the world are exploring the possibilities and potential of generative AI, and realizing the inherent complexity associated with leveraging it to address business challenges. This is why AWS made the commitment to simplify and democratize generative AI.

## Infosys Solution Overview

This solution offers a chat-like interface for users to perform semantic search and ask questions based on various internal documents within an organization. The solution is built using Amazon SageMaker JumpStart, which provides easy access to many pre-trained, public models (FMs, task-specific models) to solve a wide range of problems.

We will explain the architecture pattern and best practices to implement a knowledge base virtual assistant using AWS.

### Architecture

The following diagram depicts the architecture of the complete application.

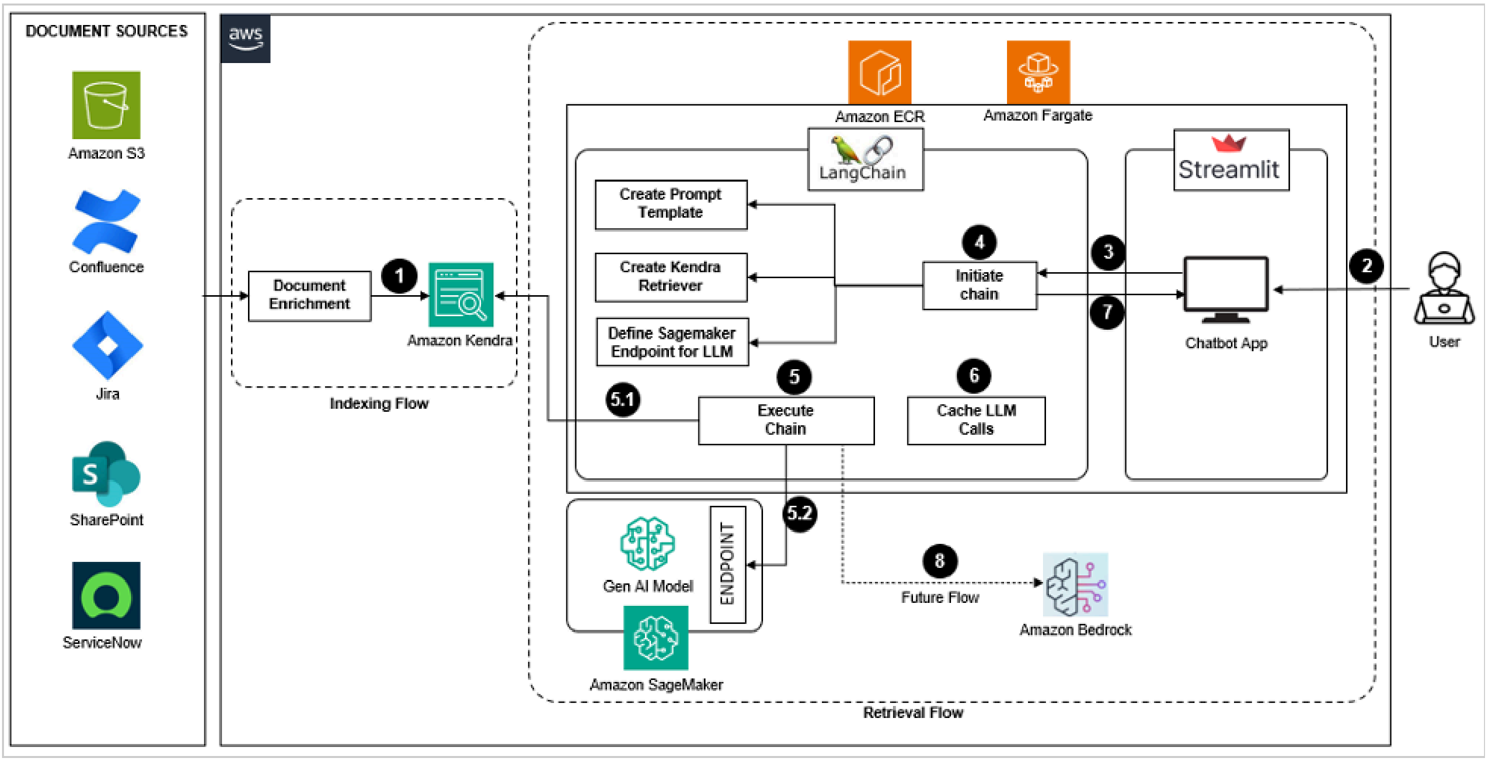

*Figure 1 – Architecture for enterprise knowledge management assistant.*

This architecture can be divided into two flows:

- **Indexing flow:** You are indexing the document contents from different sources into Amazon Kendra with optional transformations (document enrichment).
- **Retrieval flow:** As part of this, you're getting the user question, chat history, creating prompt template, getting the document excerpts, getting the final summarized answer from the LLM, and sending it back to the app.

Here are the detailed steps involved in this approach:

1. Amazon Kendra is connected to different sources of documents, which are ingested into a Kendra index along with document enrichment wherever required.
2. User asks a question in the chat interface of the application. The frontend is built using the Streamlit application and deployed under AWS Fargate, which provides a serverless container option for application deployment.
3. Chatbot application initiates the chain from a LangChain script. LangChain is a powerful open-source framework useful for developing orchestrations around LLMs. It offers many useful capabilities that are helpful for creating complex generative AI pipelines around one or more LLMs.
4. These steps are done as part of initiating chain:
    - Create Kendra Retriever.
    - Define LLM object for Amazon SageMaker endpoint.
    - Create prompt template.

- Create the chain object.

5. From the Streamlit app, execute the chain. The chain function internally does this:
    1. Query Kendra index with user question.
    2. Call SageMaker LLM endpoint by passing the user question, response from Kendra, prompt, and user chat history. This returns the final response in a summarized manner.
6. Cache the LLM response. When a user asks a question for the first time, the response from the LLM is cached. Later, when anyone asks the same question in the chatbot, it gets the response from the cache and returns to the chatbot. LangChain provides an optional caching layer for LLMs. This is useful for two reasons:
    - It can save you money by reducing the number of API calls you make to the LLM provider, if you're often requesting the same completion multiple times.
    - This would improve the performance of the application. More details can be be found here for LLM caching integrations.
7. The response from SageMaker endpoint is finally returned to the frontend application, to the user.
8. This is the future flow using Amazon Bedrock API. Bedrock makes it easy to build and scale generative AI applications with foundation models. Using the API call, you can invoke any LLMs available under Bedrock (this will replace Step 5.2 mentioned above).

## Design Principles and Components

This solution approach is based on six key design principles:

- User-friendly interface
- Cost-effective architecture
- Latest information
- Performance
- Scalability
- Privacy and security

As part of this solution, the architecture is split into five components:

- Source data ingestion
- Frontend web application
- Backend orchestration
- Application deployment
- Generative AI service

Let's go through these components in detail by ensuring the above-mentioned design principles.

- **Source data ingestion:** In this application, you are going to fetch information from a variety of document sources like Confluence, Jira, ServiceNow, SharePoint, and Amazon Simple Storage Service (Amazon S3). By using Amazon Kendra connectors, you can connect to these sources; by enabling data sync, you can ensure the freshness of the data stored in Kendra index.
- **Frontend web application:** You can use Streamlit for developing the frontend web application. Streamlit is an open-source Python-based application framework that can be used to develop rich user interface (UI) in less time. The application also stores the user chat history, and using this approach you can deploy both Streamlit-based frontend application and LangChain-based orchestration in AWS Fargate.
- **Backend orchestration:** To orchestrate the backend process, you can use LangChain framework which works as a Retrieval Augmented Generation (RAG) step in this flow. This first retrieves the document excerpts from Kendra index based on the user question, then passes it to the LLM along with the user question, prompt, and chat history to get the final response in a summarized manner. It then returns this response to the chatbot application.

  You can also enable a caching capability so you could cache the LLM calls, so similar requests can be served from the cache itself. During document ingestion into Kendra index, you can implement a document enrichment process to filter out any sensitive information like personally identifiable information (PII). By doing this, you can restrict any PII data returned as a chatbot response to the end user.
- **Application deployment:** For this, you need to create Docker image for Streamlit-based frontend application and LangChain-based scripts, and push them to Amazon Elastic Container Registry (Amazon ECR) repository. Then, create tasks in Fargate for running the application.
- **Generative AI service:** For this application, you can deploy the LLM from Amazon SageMaker JumpStart. This application uses Llama-2-7b-chat model as the LLM model which is optimized for dialogue use cases.

## Conclusion

In this post, we explored how generative AI can help you create an advanced knowledge search assistant to efficiently access your knowledge repository. By strategically utilizing a conversational interface and large language model capabilities, you can develop a sophisticated chat application to handle queries and draw insights to provide tailored, up-to-date information.

When developing applications such as this, consider basic design principles to ensure optimum productivity. The Infosys application showcased in this post can streamline workflows through automation, enable smoother knowledge sharing, elevate the learning experience, and transform information utilization.

**References:**

- Transforming aviation maintenance with the Infosys generative AI solution built on Amazon Bedrock
- How the Infosys Customer Intelligence Platform delivers a world-class customer experience
- Quickly build high-accuracy generative AI applications on enterprise data
- Exploring generative AI in conversational experiences

aws PARTNER NETWORK

Connect with Infosys ›

## Infosys – AWS Partner Spotlight

**Infosys is an AWS Premier Tier Services Partner and MSP** that enables clients to outperform competition and stay ahead of the innovation curve.

Contact Infosys | Partner Overview | Case Studies

TAGS: AWS Competency Partners, AWS MSP Partner Program, AWS Partner Guest Post, AWS Partner Solutions Architects (SA), AWS Partner Success Stories, AWS Premier Tier Services Partners, AWS Public Sector Partners, AWS Service Delivery Partners, AWS Solution Provider Partners, AWS Well-Architected Partners, Infosys, Managed Service Provider

Comments cannot be loaded… Please refresh and try again.