Creating Retrieval Augmented Generation solutions on AWS for healthcare

AWS Prescriptive Guidance

AWS Prescriptive Guidance: Creating Retrieval Augmented Generation solutions on AWS for healthcare

Table of Contents

Creating Retrieval Augmented Generation solutions on AWS for healthcare

Amazon Web Services, Accenture, and Cadiem (contributors)

March 2025 (document history)

Before large language models (LLMs) and generative AI, the task of developing automated and high-precision applications in the healthcare industry was challenging. Traditional methods relied heavily on manual data entry and analysis. The complexity of analyzing medical imaging and patient records required extensive human intervention, which often resulted in fragmented and inefficient workflows. The advancement of AI technologies helps you build hyper-personalized applications at scale. Healthcare applications can now integrate with medical knowledge bases, interpret diagnostic images with increased accuracy, and forecast patient outcomes by using predictive models.

This guide explores how LLMs are revolutionizing healthcare through Retrieval Augmented Generation applications that you can build with AWS services. *Retrieval Augmented Generation (RAG)* is a generative AI technology in which an LLM references an authoritative data source that is outside of its training data sources before generating a response. RAG applications ground the model's output in real-world knowledge, which reduces hallucinations and increases response relevance. In the healthcare sector, RAG can be used to provide accurate and up-to-date medical information, ensuring that healthcare providers have access to the latest research and clinical guidelines. By transforming data into actionable insights and automating complex processes, these technologies help enhance patient care, streamline operations, and improve productivity of healthcare professionals.

In Amazon Bedrock, you can fine-tune LLMs and integrate them with intelligent agents to create advanced healthcare solutions. Highlighting the synergy between Amazon OpenSearch Service and Amazon Neptune, the guide demonstrates how these services elevate RAG solutions through enhanced search relevance and advanced multi-source data retrieval. You can orchestrate comprehensive Amazon Bedrock solutions that use Amazon Bedrock agents and LangChain to seamlessly coordinate interactions across diverse data repositories. This integration demonstrates the power of combining specialized services to create more effective and efficient AI-driven systems.

Patient care and productivity

This guide presents two real-world use cases for patient care and productivity: patient data augmentation and predicting re-admission risks. It provides strategic blueprints for implementing these solutions at scale, offering healthcare organizations a clear path to industrializing AI-driven processes. Through these insights, healthcare institutions can use advanced AI technologies to create more efficient, intelligent workflows.

Talent management

This guide also outlines strategies for re-skilling and empowering healthcare workers to seamlessly integrate generative AI into their daily routines. This can enhance both productivity and patient care quality. By equipping their workforce with the skills to effectively use advanced AI tools, healthcare organizations can maximize their return on investment and drive innovation in patient care.

This AI-powered talent management solution includes the following key features:

- **Intelligent talent resume parser** – By using the advanced LLMs available in Amazon Bedrock, this tool efficiently extracts and analyzes critical talent skills and attributes from resumes. This tool can streamline the recruitment process.
- **Talent knowledge base** – Powered by Amazon Neptune, this dynamic database provides real-time insights into staffing levels, skill distribution, and industry trends. This helps you make data-driven decisions about workforce management.
- **Learning recommendation engine** – This AI-driven tool identifies skill gaps within the organization and recommends personalized training programs for medical staff. This tool promotes continuous professional development and helps your workforce adapt to evolving healthcare technologies.

Together, these AI-driven features help optimize workforce performance, revolutionizing talent management with increased intelligence and efficiency.

Opportunities and challenges

Amazon Bedrock can provide enhanced productivity, scalability, cost-effectiveness, and data-driven insights. Amazon Bedrock empowers healthcare organizations to use LLMs effectively across various use cases, from content creation and data analysis to automated decision-making. This guide provides approaches for overcoming common generative AI challenges, such as data quality issues, infrastructure scalability, maintenance of model performance, and continuous improvement requirements during the transition from proof of concept to production.

Opportunities for generative AI applications in healthcare

The healthcare industry is poised for a transformative shift, driven by the opportunities presented by generative AI applications. Generative AI has the potential to enhance patient care, streamline operations, and accelerate medical research. By using advanced AI models, healthcare providers can automate the augmentation of medical records. Comprehensive and up-to-date patient histories facilitate more accurate diagnoses and treatment plans. AI-driven image analysis, such as interpreting sonograms and other medical imaging, can provide rapid and precise insights, reducing the workload on medical professionals and minimizing the risk of human error.

Beyond diagnostics and treatment, generative AI can play a pivotal role in predictive analytics. Predictive analytics helps healthcare organizations anticipate patient outcomes and personalize care plans accordingly. This technology can also optimize administrative processes, from managing patient data to streamlining communication between providers and patients. By integrating generative AI solutions with existing healthcare systems, medical institutions can achieve greater efficiency, reduce costs, and ultimately deliver higher quality care. The integration of AI with healthcare is not just an enhancement but a fundamental shift towards more intelligent, responsive, and patient-centric care.

Advanced image analysis

Combining Amazon Bedrock with data stores, such as Amazon Neptune and Amazon OpenSearch Service, can help you address the complexities of advanced image analysis in healthcare. Information retrieval solutions can augment the disease discovery process and enhance interpretation accuracy by assessing diagnostic images and interpreting sonograms. The solution can integrate the visual and textual assessment data with manual patient assessment review by doctors.

Challenges with industrializing the solutions

The primary obstacles to tackle when industrializing AI solutions in healthcare is data quality and availability. Healthcare data often exists in fragmented, inconsistent formats. Making sure that AI models have access to clean, structured, and representative data is crucial for maintaining performance in real-world scenarios. Infrastructure scalability can become a challenge because production environments. These environments need to handle large volumes of real-time patient data while providing fast response times and maintaining compliance with data privacy regulations, such as Health Insurance Portability and Accountability Act (HIPAA). Moreover, with emerging medical information and patient data that evolves over time, AI models need to be retrained and updated to stay relevant and give accurate recommendations. Finally, integrating these AI solutions into existing healthcare systems can be complex due to interoperability issues and the need for alignment with current clinical workflows. This integration requires both technical and operational changes.

Use case: Building a medical intelligence application with augmented patient data

Generative AI can help augment patient care and staff productivity by enhancing both clinical and administrative functions. AI-driven image analysis, such as interpreting sonograms, accelerates diagnostic processes and improves accuracy. It can provide critical insights that support timely medical interventions.

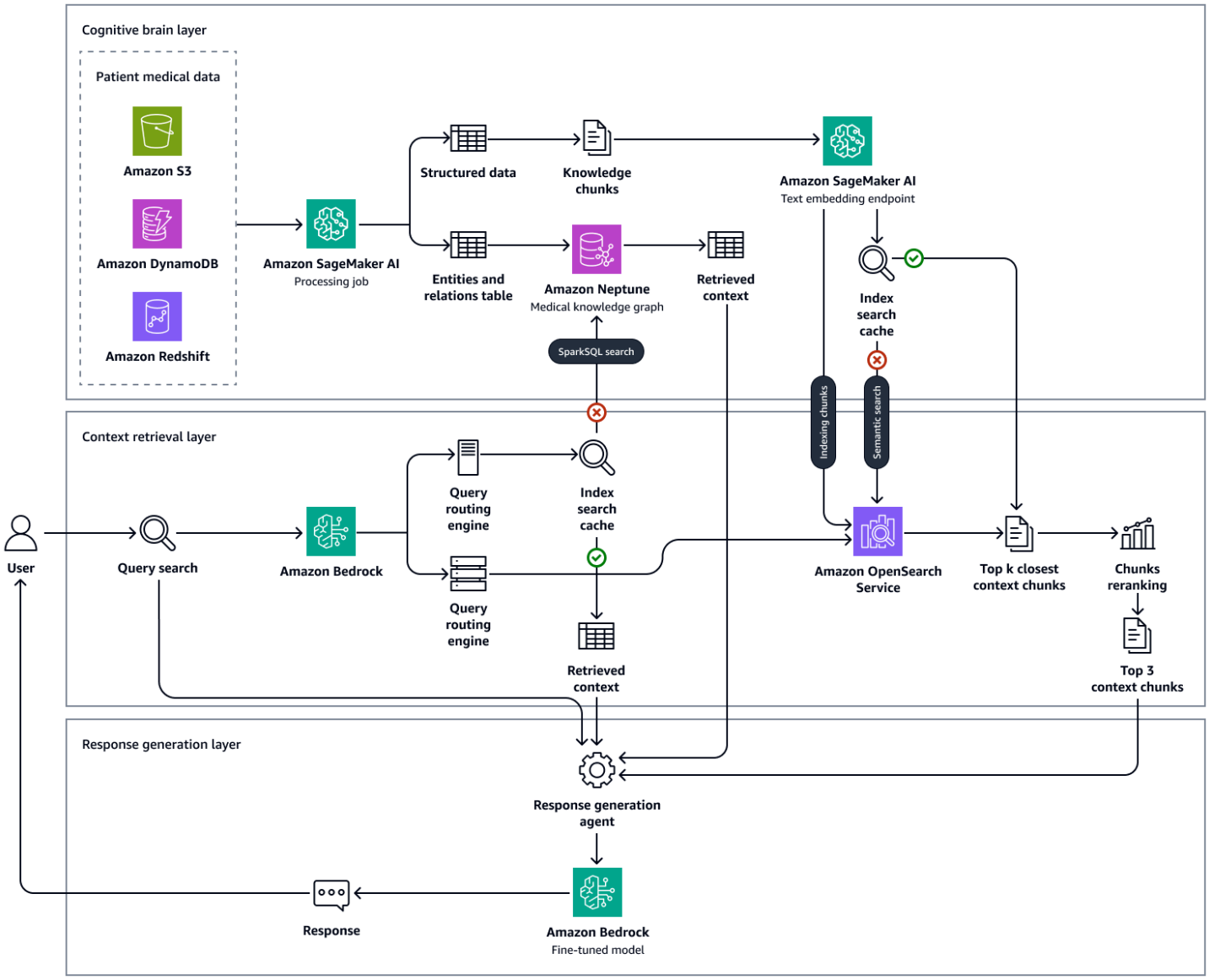
When you combine generative AI models with knowledge graphs, you can automate the chronological organization of electronic patient records. This helps you integrate real-time data from doctor-patient interactions, symptoms, diagnoses, lab results, and image analysis. This equips the doctor with comprehensive patient data. This data helps the doctor make more accurate and timely medical decisions, enhancing both patient outcomes and healthcare provider productivity.

Solution overview

AI can empower doctors and clinicians by synthesizing patient data and medical knowledge to provide valuable insights. This Retrieval Augmented Generation (RAG) solution is a medical intelligence engine that consumes a comprehensive set of patient data and knowledge from millions of clinical interactions. It harnesses the power of generative AI to create evidence-based insights for improved patient care. It is designed to enhance the clinical workflows, reduce errors, and improve patient outcomes.

The solution includes an automated image-processing capability that is powered by LLMs. This capability reduces the amount of time that medical personnel must spend manually searching for similar diagnostic images and analyzing diagnostic results.

The following image shows the end-to-end-workflow for this solution. It uses Amazon Neptune, Amazon SageMaker AI, Amazon OpenSearch Service, and a foundation model in Amazon Bedrock. For the context retrieval agent that interacts with the medical knowledge graph in Neptune, you can choose between an Amazon Bedrock agent and a LangChain agent.



In our experiments with sample medical questions, we observed that the final responses generated by our approach using knowledge graph maintained in Neptune, OpenSearch vector database housing clinical knowledge base, and Amazon Bedrock LLMs were grounded in factuality and are far more accurate by reducing the false positives and boosting the true positives. This solution can generate evidence-based insights on patient's health status and aims to enhance the clinical workflows, reduce errors, and improve patient outcomes.

Building this solution consists of the following steps:

- Step 1: Discovering data
- Step 2: Building a medical knowledge graph

- Step 3: Building context retrieval agents to query the medical knowledge graph
- Step 4: Creating a knowledge base of real-time, descriptive data
- Step 5: Using LLMs to answer medical questions

Step 1: Discovering data

There are many open source medical datasets that you can use to support the development of a healthcare AI-driven solution. One such dataset is the MIMIC-IV dataset, which is a publicly available electronic health record (EHR) dataset that is widely used in the healthcare research community. MIMIC-IV contains detailed clinical information, including free-text discharge notes from patient records. You can use these records to experiment with text summation and entity extraction techniques. These techniques help you extract medical information (such as patient symptoms, administered medications, and prescribed treatments) from unstructured text.

You might also use a dataset that provides annotated, de-identified patient discharge summaries that are specifically curated for research purposes. A discharge summary dataset can help you experiment with entity extraction, allowing you to identify key medical entities (such as conditions, procedures, and medications) from the text. Step 2: Building a medical knowledge graph in this guide describes how you can use the structured data extracted from the MIMIC-IV and discharge summary datasets to create a medical knowledge graph. This medical knowledge graph serves as the backbone for advanced querying and decision-support systems for healthcare professionals.

In addition to text-based datasets, you can use image datasets. For example, the Musculoskeletal Radiographs (MURA) dataset, which is a comprehensive database of multi-view radiographic images of bones. Use such image datasets to experiment with diagnostic assessment through medical image decoding techniques. These decoding techniques are crucial for early diagnosis of diseases, such as musculoskeletal diseases, cardiovascular diseases, and osteoporosis. By fine-tuning vision and language foundation models on the medical image dataset, you can detect of abnormalities in diagnostic images. This helps the system provide clinicians with early and accurate diagnostic insights. By using image and text datasets, you can create an AI-driven healthcare application that is capable of processing both text and image data to improve patient care.

Step 2: Building a medical knowledge graph

For any healthcare organization that wants to build a decision-support system based on a massive knowledge base, a key challenge is to locate and extract the medical entities that are present in

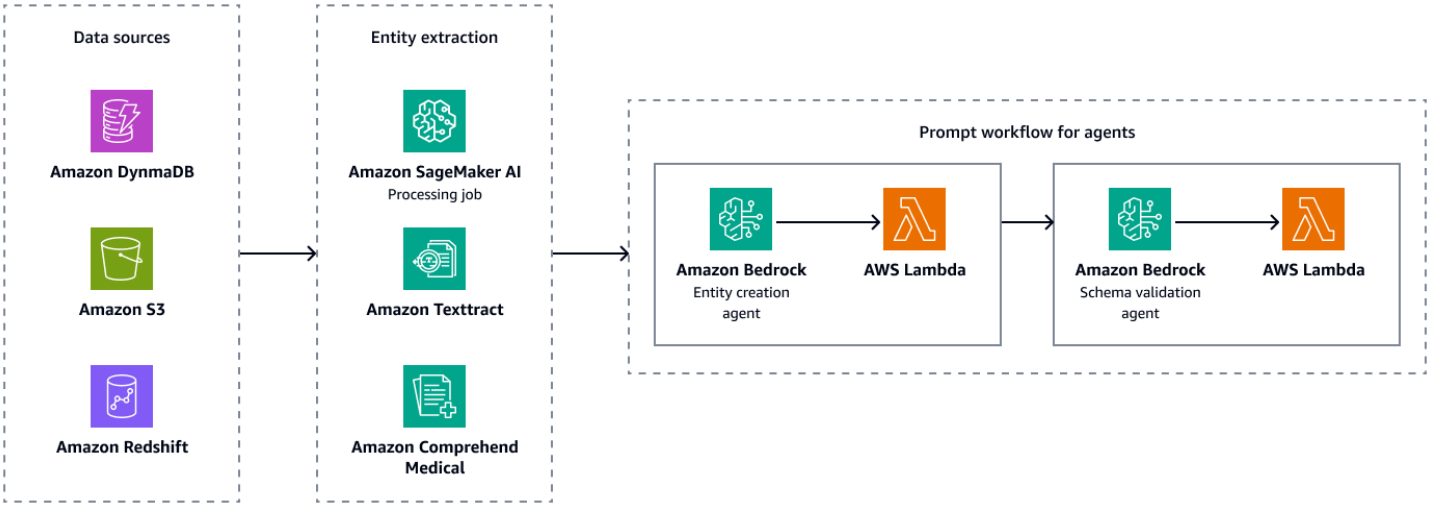
the clinical notes, medical journals, discharge summaries, and other data sources. You also need to capture the temporal relationships, subjects, and certainty assessments from these medical records in order to effectively use the extracted entities, attributes, and relationships.

The first step is to extract medical concepts from the unstructured medical text by using a few-shot prompt for a foundation model, such as Llama 3 in Amazon Bedrock. *Few-shot prompting* is when you provide an LLM with a small number of examples that demonstrate the task and desired output before asking it to perform a similar task. Using an LLM-based medical entity extractor, you can parse the unstructured medical text and then generate a structured data representation of the medical knowledge entities. You can also store the patient attributes for downstream analysis and automation. The entity extraction process includes the following actions:

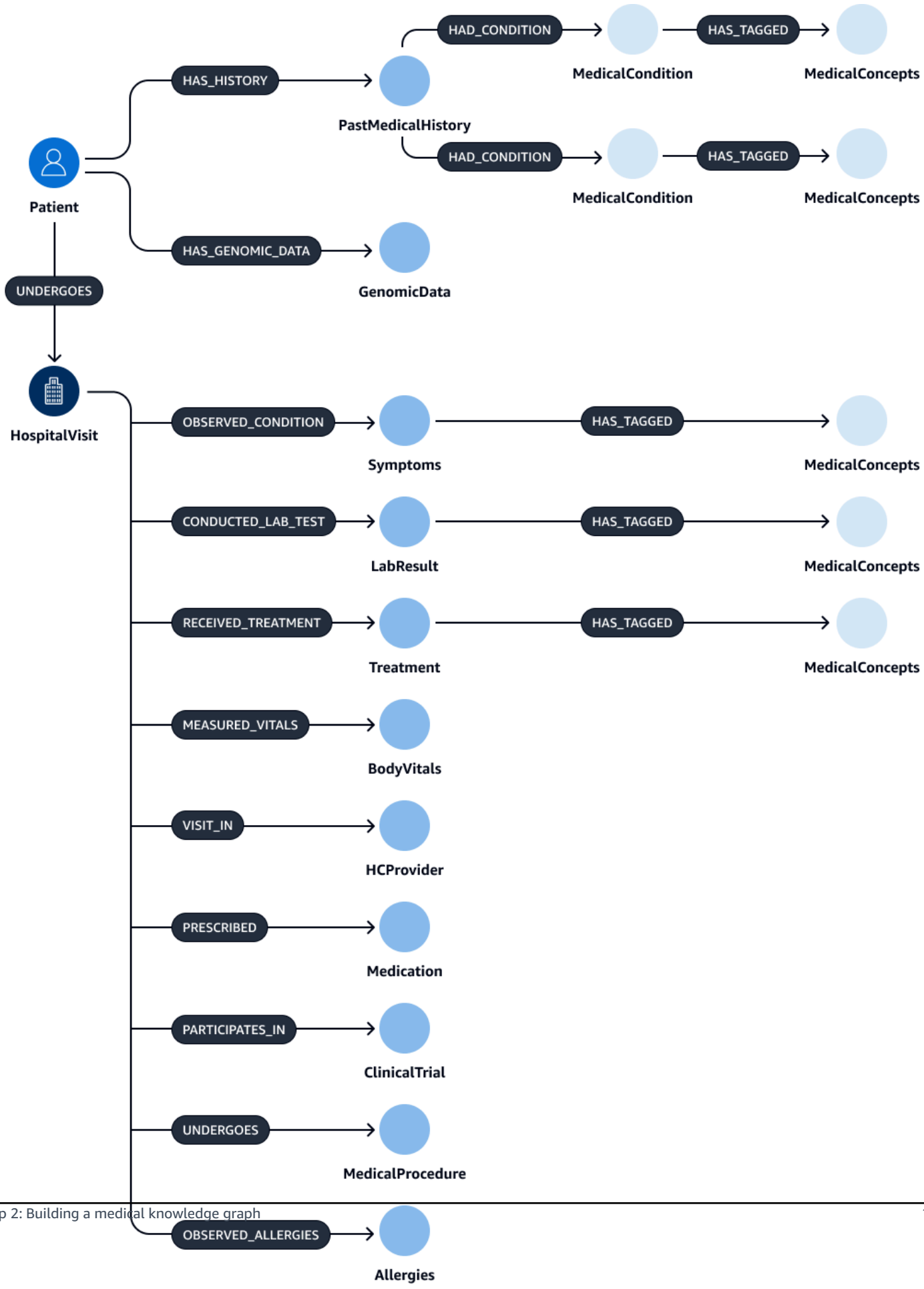
- Extract information about medical concepts, such as diseases, medications, medical devices, dosage, medicine frequency, medicine duration, symptoms, medical procedures, and their clinically relevant attributes.
- Capture functional features, such as temporal relationships between extracted entities, subjects, and certainty assessments.
- Expand standard medical vocabularies, such as the following:
  - Concept identifiers (RxCUI) from the RxNorm database
  - Codes from the International Classification of Diseases, 10th Revision, Clinical Modification (ICD-10-CM)
  - Terms from Medical Subject Headings (MeSH)
  - Concepts from Systematized Nomenclature of Medicine, Clinical Terms (SNOMED CT)
  - Codes from Unified Medical Language System (UMLS)
- Summarize discharge notes and derive medical insights from transcripts.

The following figure shows the entity extraction and schema validation steps to create valid paired combinations of entities, attributes, and relationships. You can store unstructured data, such as discharge summaries or patient notes, in Amazon Simple Storage Service (Amazon S3). You can store structured data, such as enterprise resource planning (ERP) data, electronic patient records, and lab information systems, in Amazon Redshift and Amazon DynamoDB. You can build an Amazon Bedrock entity-creation agent. This agent can integrate services, such as Amazon SageMaker AI data-extraction pipelines, Amazon Textract, and Amazon Comprehend Medical, to extract entities, relationships, and attributes from the structured and unstructured data sources. Finally, you use an Amazon Bedrock schema-validation agent to make sure that the extracted

entities and relationships conform to the predefined graph schema and maintain the integrity of node-edge connections and associated properties.



After extraction and validation of the entities, relations, and attributes, you can link them to create a subject-object-predicate triplet. You ingest this data into an Amazon Neptune graph database, as shown in the following figure. Graph databases are optimized to store and query the relationships between data items.


HAD_CONDITION
HAS_TAGGED
MedicalCondition
MedicalConcepts
HAS_HISTORY
PastMedicalHistory
HAD_CONDITION
HAS_TAGGED
MedicalCondition
MedicalConcepts
Patient
HAS_GENOMIC_DATA
GenomicData
UNDERGOES
HospitalVisit
OBSERVED_CONDITION
HAS_TAGGED
Symptoms
MedicalConcepts
CONDUCTED_LAB_TEST
HAS_TAGGED
LabResult
MedicalConcepts
RECEIVED_TREATMENT
HAS_TAGGED
Treatment
MedicalConcepts
MEASURED_VITALS
BodyVitals
VISIT_IN
HCProvider
PRESCRIBED
Medication
PARTICIPATES_IN
ClinicalTrial
UNDERGOES
MedicalProcedure
OBSERVED_ALLERGIES
Allergies

You could create a comprehensive knowledge graph with this data. A knowledge graph helps you organize and query all kinds of connected information. For example, you might create a knowledge graph that has the following major nodes: `HospitalVisit`, `PastMedicalHistory`, `Symptoms`, `Medication`, `MedicalProcedures`, and `Treatment`.

The following tables lists the entities and their attributes that you might extract from discharge notes.

Entity	Attributes
`Patient`	`PatientID` , `Name`, `Age`, `Gender`, `Address`, `ContactInformation`
`HospitalVisit`	`VisitDate` , `Reason`, `Notes`
`HealthcareProvider`	`ProviderID` , `Name`, `Specialty` , `ContactInformation` , `Address`, `AffiliatedInstitution`
`Symptoms`	`Description` , `RiskFactors`
`Allergies`	`AllergyType` , `Duration`
`Medication`	`MedicationID` , `Name`, `Description` , `Dosage`, `SideEffects` , `Manufacturer`
`PastMedicalHistory`	`ContinuingMedicines`
`MedicalCondition`	`ConditionName` , `Severity`, `Treatment Received` , `DoctorinCharge` , `HospitalName` , `MedicinesFollowed`
`BodyVitals`	`HeartRate` , `BloodPressure` , `RespiratoryRate` , `BodyTemperature` , `BMI`
`LabResult`	`LabResultID` , `PatientID` , `TestName`, `Result`, `Date`

Entity	Attributes
ClinicalTrial	TrialID, Name, Description , Phase, Status, StartDate , EndDate
GenomicData	GenomicDataID , PatientID , SequenceData , VariantInformation
Treatment	TreatmentID , Name, Description , Type, SideEffects
MedicalProcedure	ProcedureID , Name, Description , Risks, Outcomes
MedicalConcepts	UMLSCodes , MedicalVocabularies

The following table lists the relationships that entities might have and their corresponding attributes. For example, the Patient entity might connect to the HospitalVisit entity with the [UNDERGOES] relationship. The attribute for this relationship is VisitDate.

Subject entity	Relationship	Object entity	Attributes
Patient	[UNDERGOES]	HospitalVisit	VisitDate
HospitalVisit	[VISIT_IN]	HealthcareProvider	ProviderName , Location, ProviderID , VisitDate
HospitalVisit	[OBSERVED_CONDITION]	Symptoms	Severity, CurrentStatus , VisitDate
HospitalVisit	[RECEIVED_TREATMENT]	Treatment	Duration, Dosage, VisitDate

Subject entity	Relationship	Object entity	Attributes
HospitalVisit	[PRESCRIBED]	Medication	Duration, Dosage, Adherence , VisitDate
Patient	[HAS_HISTORY]	PastMedicalHistory	None
PastMedicalHistory	[HAD_CONDITION]	MedicalCondition	DiagnosisDate , CurrentStatus
HospitalVisit	[PARTICIPATES_IN]	ClinicalTrial	VisitDate , Status, Outcomes
Patient	[HAS_GENOMIC_DATA]	GenomicData	CollectionDate
HospitalVisit	[OBSERVED_ALLERGIES]	Allergies	VisitDate
HospitalVisit	[CONDUCTED_LAB_TEST]	LabResult	VisitDate , AnalysisDate , Interpretation
HospitalVisit	[UNDERGOES]	MedicalProcedure	VisitDate , Outcome
MedicalCondition	[HAS_TAGGED]	MedicalConcepts	None
LabResult	[HAS_TAGGED]	MedicalConcepts	None
Treatment	[HAS_TAGGED]	MedicalConcepts	None
Symptoms	[HAS_TAGGED]	MedicalConcepts	None

Step 3: Building context retrieval agents to query the medical knowledge graph

After you build the medical graph database, the next step is to build agents for graph interaction. The agents retrieve the correct and required context for the query that a doctor or clinician inputs. There are several options for configuring these agents that retrieve the context from the knowledge graph:

- Amazon Bedrock agents
- LangChain agents

Amazon Bedrock agents for graph interaction

Amazon Bedrock agents work seamlessly with Amazon Neptune graph databases. You can perform advanced interactions through Amazon Bedrock action groups. The action group initiates the process by calling an AWS Lambda function, which runs Neptune openCypher queries.

For querying a knowledge graph, you can use two distinct approaches: direct query execution or querying with context embedding. These approaches can be applied independently or combined, depending on your specific use case and ranking criteria. By combining both approaches, you can provide more comprehensive context to the LLM, which can improve results. The following are the two query execution approaches:

- **Direct Cypher query execution without embeddings** – The Lambda function executes queries directly against Neptune without any embeddings-based search. The following is an example of this approach:

```
MATCH (p:Patient)-[u:UNDERGOES]->(h:HospitalVisit) WHERE h.Reason = 'Acute Diabetes'
 AND date(u.VisitDate) > date('2024-01-01')
RETURN p.PatientID, p.Name, p.Age, p.Gender, p.Address, p.ContactInformation
```

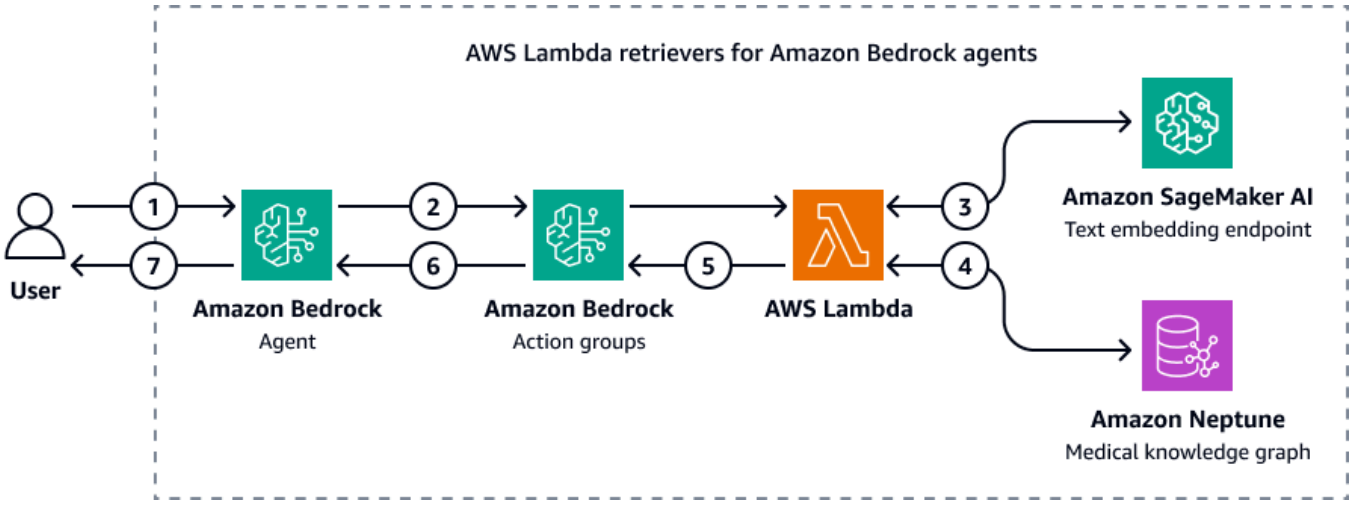
- **Direct Cypher query execution using embedding search** – The Lambda function uses embedding search to enhance the query results. This approach enhances the query execution by incorporating *embeddings*, which are dense vector representations of data. Embeddings are particularly useful when the query requires semantic similarity or broader understanding beyond exact matches. You can use pre-trained or custom-trained models to generate embeddings for each medical condition. The following is an example of this approach:

```
CALL { WITH "Acute Diabetes" AS query_term RETURN search_embedding(query_term) AS
 similar_reasons }

MATCH (p:Patient)-[u:UNDERGOES]->(h:HospitalVisit) WHERE h.Reason IN similar reasons
 AND date(u.VisitDate) > date('2024-01-01')
RETURN p.PatientID, p.Name, p.Age, p.Gender, p.Address, p.ContactInformation
```

In this example, the `search_embedding("Acute Diabetes")` function retrieves conditions that are semantically close to "Acute Diabetes." This helps the query to also find patients that have conditions such as pre-diabetes or metabolic syndrome.

The following image shows how Amazon Bedrock agents interact with Amazon Neptune in order to perform a Cypher query of a medical knowledge graph.



The diagram shows the following workflow:

1. The user submits a question to the Amazon Bedrock agent.
2. The Amazon Bedrock agent passes the question and input filter variables to the Amazon Bedrock action groups. These action groups contain an AWS Lambda function that interacts with the Amazon SageMaker AI text embedding endpoint and the Amazon Neptune medical knowledge graph.
3. The Lambda function integrates with the SageMaker AI text embedding endpoint to perform a semantic search within the openCypher query. It converts the natural language query into an openCypher query by using underlying LangChain agents.

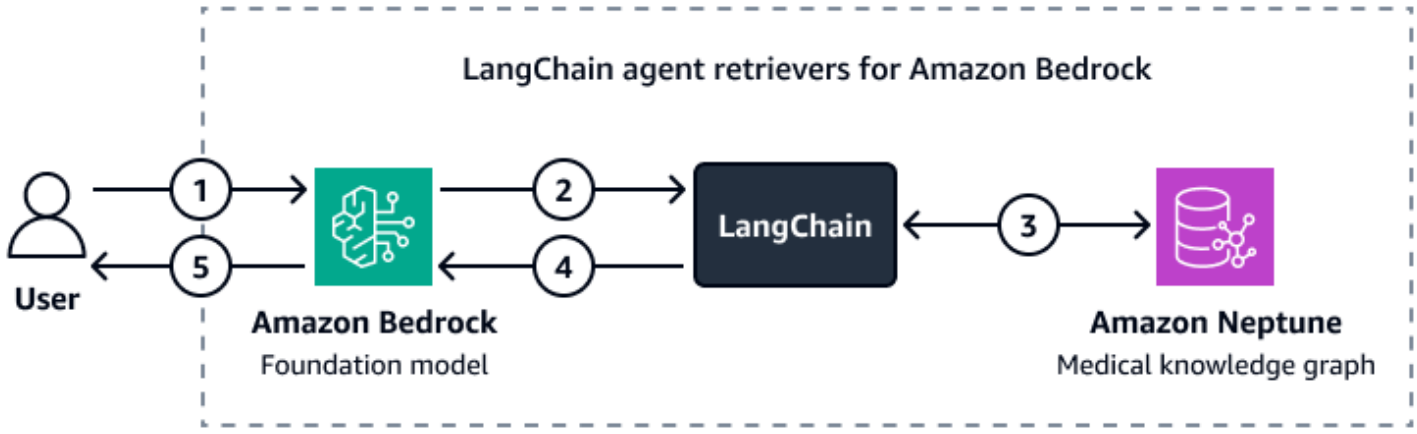
4. The Lambda function queries the Neptune medical knowledge graph for the correct dataset and receives the output from the Neptune medical knowledge graph.
5. The Lambda function returns the results from Neptune to the Amazon Bedrock action groups.
6. The Amazon Bedrock action groups send the retrieved context to the Amazon Bedrock agent.
7. The Amazon Bedrock agent generates the response by using the original user query and the retrieved context from the knowledge graph.

LangChain agents for graph interaction

You can integrate LangChain with Neptune to enable graph-based queries and retrievals. This approach can enhance AI-driven workflows by using the graph database capabilities in Neptune. The custom LangChain retriever acts as an intermediary. The foundational model in Amazon Bedrock can interact with Neptune by using both direct Cypher queries and more complex graph algorithms.

You can use the custom retriever to refine how the LangChain agent interacts with the Neptune graph algorithms. For example, you can use few-shot prompting, which helps you tailor the foundation model's responses based on specific patterns or examples. You can also apply LLM-identified filters to refine the context and improve the precision of responses. This can improve the efficiency and accuracy of the overall retrieval process when interacting with complex graph data.

The following image shows how a custom LangChain agent orchestrates the interaction between an Amazon Bedrock foundation model and an Amazon Neptune medical knowledge graph.



The diagram shows the following workflow:

1. A user submits a question to Amazon Bedrock and the LangChain agent.

2. The Amazon Bedrock foundation model uses the Neptune schema, which is provided by the LangChain agent, to generate a query for the user's question.
3. The LangChain agent runs the query against the Amazon Neptune medical knowledge graph.
4. The LangChain agent sends the retrieved context to the Amazon Bedrock foundation model.
5. The Amazon Bedrock foundation model uses the retrieved context to generate an answer to the user's question.

Step 4: Creating a knowledge base of real-time, descriptive data

Next, you create a knowledge base of real-time, descriptive doctor-patient interaction notes, diagnostic image assessments, and lab analysis reports. This knowledge base is a vector database. By using a vector database, which can store descriptive medical knowledge in an indexed, vectorized form, healthcare providers can efficiently query and access relevant information from a vast repository. These vectorized representations help you retrieve semantically similar data. Care providers can quickly navigate through clinical notes, medical images, and lab results. This accelerates informed decision-making by offering instant access to contextually relevant information, enhancing the accuracy and speed of diagnoses and treatment plans.

Using an OpenSearch Service medical knowledge base

Amazon OpenSearch Service can manage large volumes of high-dimensional medical data. It is a managed service that facilitates high-performance search and real-time analytics. It is well suited as a vector database for RAG applications. OpenSearch Service acts as a backend tool to manage vast amounts of unstructured or semi-structured data, such as medical records, research articles, and clinical notes. Its advanced semantic search capabilities help you retrieve contextually relevant information. This makes it particularly useful in applications such as clinical decision-support systems, patient query resolution tools, and healthcare knowledge management systems. For instance, a clinician can quickly find relevant patient data or research studies that match specific symptoms or treatment protocols. This helps clinicians make decisions that are informed by the most up-to-date and relevant information.

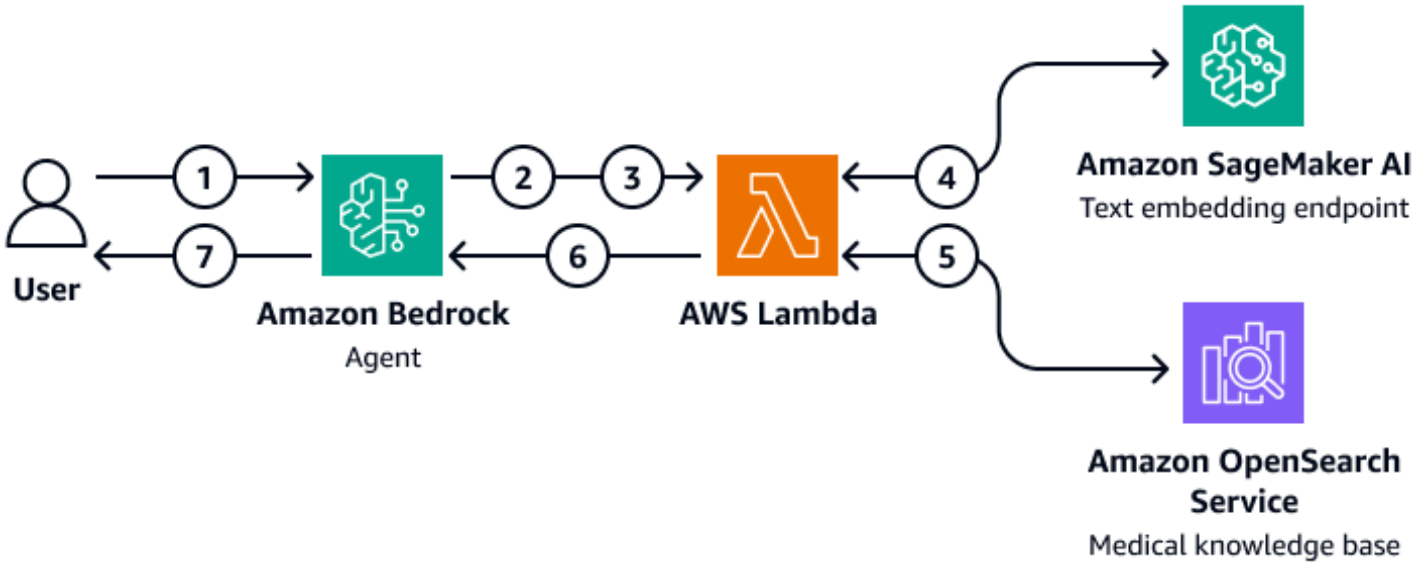
OpenSearch Service can scale and handle real-time data indexing and querying. This makes it ideal for dynamic healthcare environments where timely access to accurate information is critical. Additionally, it has multi-modal search capabilities that are optimal for searches that require

multiple inputs, such as medical images and doctor notes. When implementing OpenSearch Service for healthcare applications, it is crucial that you define precise fields and mappings in order to optimize data indexing and retrieval. *Fields* represent the individual pieces of data, such as patient records, medical histories, and diagnostic codes. *Mappings* define how these fields are stored (in embedding form or original form) and queried. For healthcare applications, it is essential to establish mappings that accommodate various data types, including structured data (such as numerical test results), semi-structured data (such as patient notes), and unstructured data (such as medical images)

In OpenSearch Service, you can perform full-text neural search queries through curated prompts to search through medical records, clinical notes, or research papers to quickly find relevant information about specific symptoms, treatments, or patient histories. Neural search queries automatically handle the embedding of the input prompt and images by using built-in neural network models. This helps it understand and capture the deeper semantic relationships in multi-modal data, offering more context-aware and precise search results compared to other search query algorithms, such as like k-Nearest Neighbor (k-NN) search.

Creating a RAG architecture

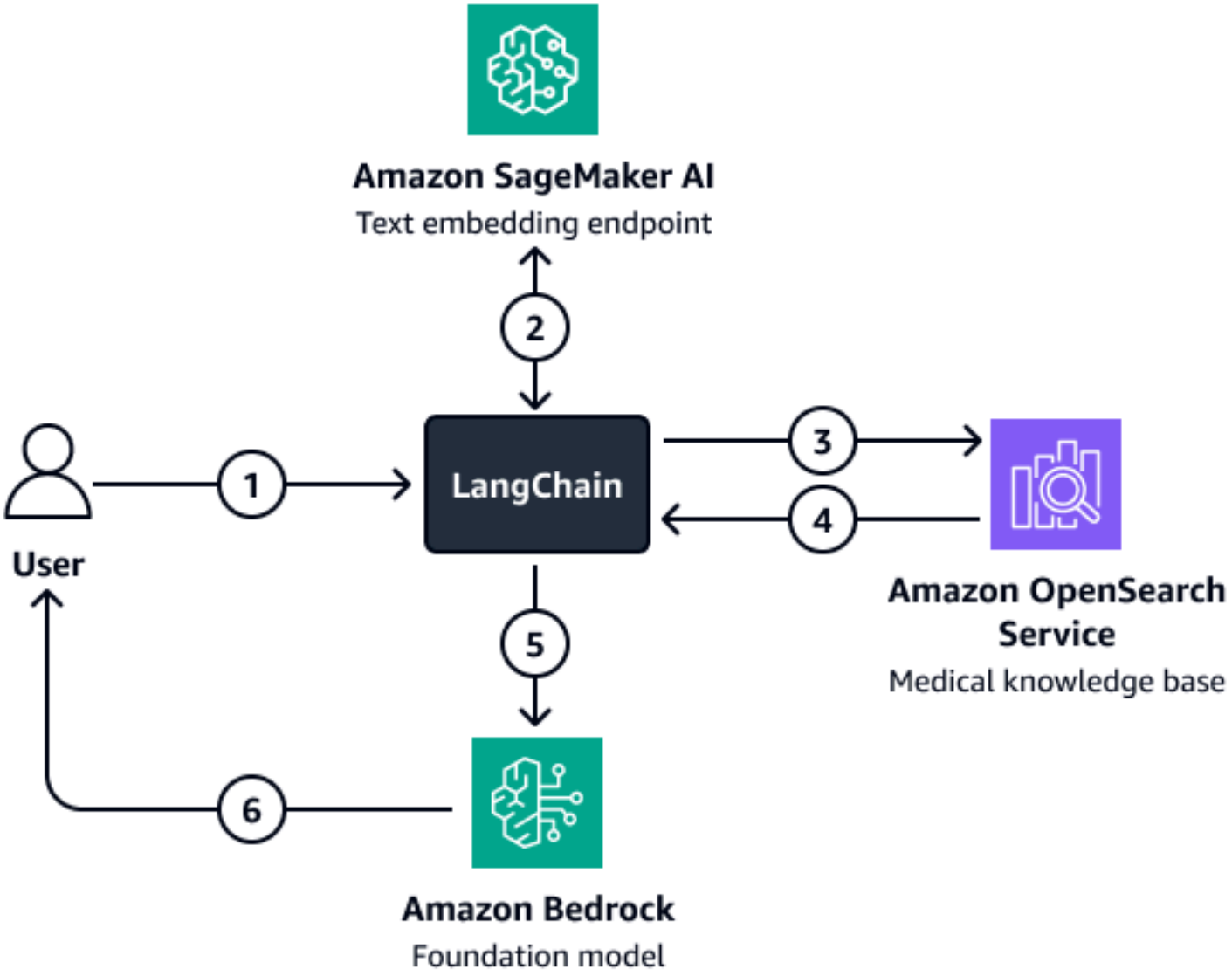
You can deploy a customized RAG solution that uses Amazon Bedrock agents to query a medical knowledge base in OpenSearch Service. To accomplish this, you create an AWS Lambda function that can interact with and query OpenSearch Service. The Lambda function embeds the user's input question by accessing a SageMaker AI text embedding endpoint. The Amazon Bedrock agent passes additional query parameters as inputs to the Lambda function. The function queries the medical knowledge base in OpenSearch Service, which returns the relevant medical content. After you set up the Lambda function, add it as an action group within the Amazon Bedrock agent. The Amazon Bedrock agent takes the user's input, identifies the necessary variables, passes the variables and question to the Lambda function, and then initiates the function. The function returns a context that helps the foundation model provide a more accurate answer to the user's question.



The diagram shows the following workflow:

1. A user submits a question to the Amazon Bedrock agent.
2. The Amazon Bedrock agent selects which action group to initiate.
3. The Amazon Bedrock agent initiates an AWS Lambda function and passes parameters to it.
4. The Lambda function initiates the Amazon SageMaker AI text embedding model to embed the user question.
5. The Lambda function passes the embedded text and additional parameters and filters to Amazon OpenSearch Service. Amazon OpenSearch Service queries the medical knowledge base and returns results to the Lambda function.
6. The Lambda function passes the results back to the Amazon Bedrock agent.
7. The foundation model in the Amazon Bedrock agent generates a response based on the results and returns the response to the user.

For situations where more complex filtering is involved, you can use a custom LangChain retriever. Create this retriever by setting up an OpenSearch Service vector search client that is loaded directly into LangChain. This architecture allows you to pass more variables in order to create the filter parameters. After the retriever is set up, use the Amazon Bedrock model and retriever to set up a retrieval question-answering chain. This chain orchestrates the interaction between the model and retriever by passing the user input and potential filters to the retriever. The retriever returns relevant context that helps the foundation model answer the user's question.



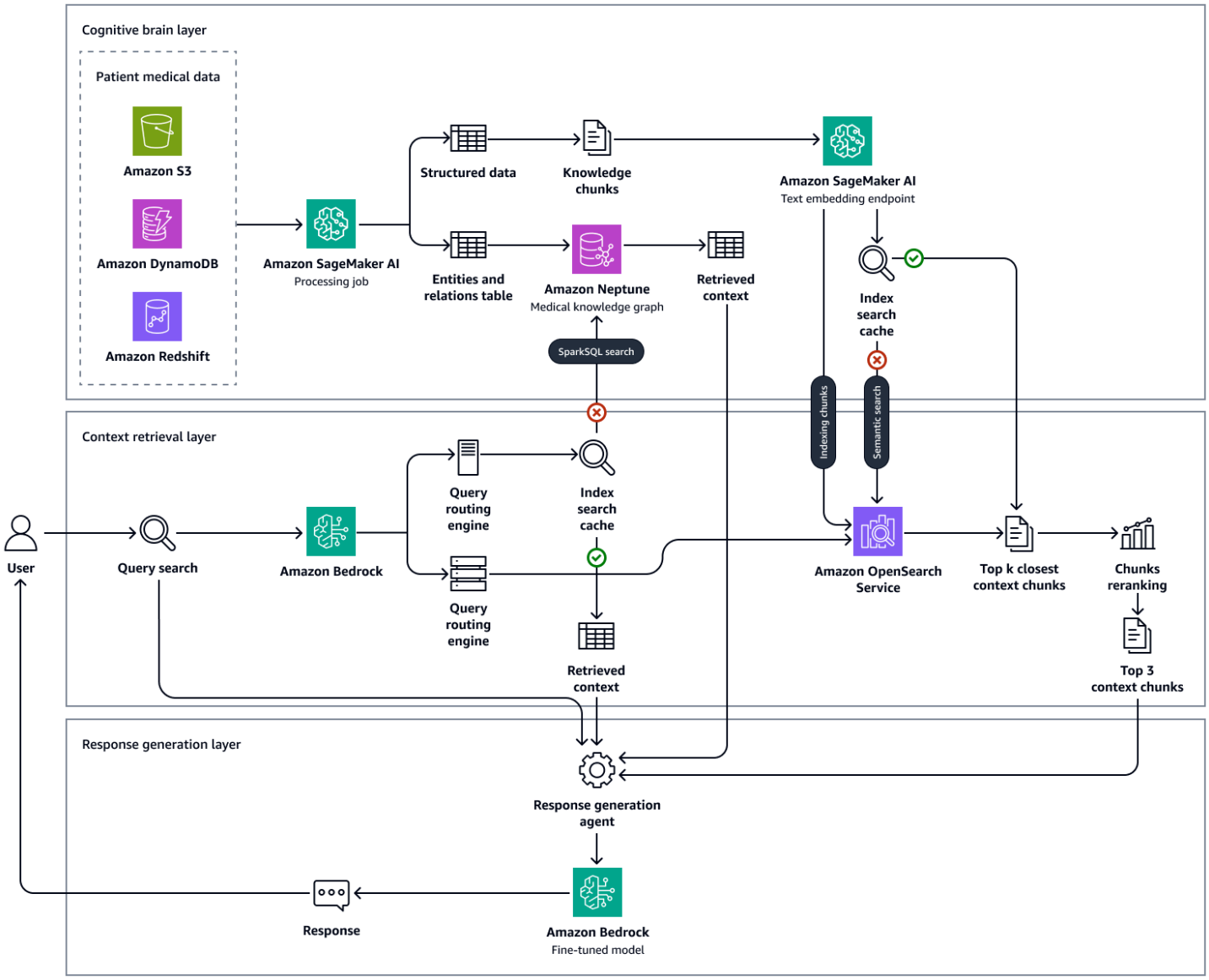
The diagram shows the following workflow:

1. A user submits a question to the LangChain retriever agent.
2. The LangChain retriever agent sends the question to the Amazon SageMaker AI text embedding endpoint to embed the question.
3. The LangChain retriever agent passes the embedded text to Amazon OpenSearch Service.
4. Amazon OpenSearch Service returns the retrieved documents to the LangChain retriever agent.
5. The LangChain retriever agent passes the user question and retrieved context to the Amazon Bedrock foundation model.
6. The foundation model generates a response and sends it to the user.

Step 5: Using LLMs to answer medical questions

The previous steps help you build a medical intelligence application that can fetch a patient's medical records and summarize relevant medications and potential diagnoses. Now, you build the generation layer. This layer uses the generative capabilities of an LLM in Amazon Bedrock, such as Llama 3, to augment the application's output.

When a clinician inputs a query, the *context retrieval layer* of the application performs the retrieval process from the knowledge graph and returns the top records that pertain to the patient's history, demographics, symptoms, diagnosis, and outcomes. From the vector database, it also retrieves real-time, descriptive doctor-patient interaction notes, diagnostic image assessment insights, lab analysis report summaries, and insights from huge corpus of medical research and academic books. These top retrieved results, the clinician's query, and the prompts (which are tailored to curate answers based on the nature of the query), are then passed to the foundation model in Amazon Bedrock. This is the *response generation layer*. The LLM uses the retrieved context to generate a response to the clinician's query. The following figure shows the end-to-end workflow of the steps in this solution.



You can use a pre-trained foundational model in Amazon Bedrock, such as Llama 3, for a range of use cases that the medical intelligence application has to handle. The most effective LLM for a given task varies depending on the use case. For example, a pre-trained model might be sufficient to summarize patient-doctor conversations, search through medications and patient histories, and retrieve insights from internal medical data sets and bodies of scientific knowledge. However, a fine-tuned LLM might be necessary for other complex use cases, such as real-time laboratory evaluations, medical procedure recommendations, and predictions of patient outcomes. You can fine-tune an LLM by training it on medical domain datasets. Specific or complex healthcare and life sciences requirements drive development of these fine-tuned models.

For more information about fine-tuning an LLM or choosing an existing LLM that has been trained on medical domain data, see Using large language models for healthcare and life science use cases.

Alignment to the AWS Well-Architected Framework

The solution aligns with all six pillars of the AWS Well-Architected Framework as follows:

- **Operational excellence** – The architecture is decoupled for efficient monitoring and updates. Amazon Bedrock agents and AWS Lambda help you quickly deploy and roll back tools.
- **Security** – This solution is designed to comply with healthcare regulations, such as HIPAA. You can also implement encryption, fine-grained access control, and Amazon Bedrock guardrails to help protect patient data.
- **Reliability** – AWS managed services, such as Amazon OpenSearch Service and Amazon Bedrock, provide the infrastructure for continuous model interaction.
- **Performance efficiency** – The RAG solution retrieves relevant data quickly by using optimized semantic search and Cypher queries, while an agent router identifies optimal routes for user queries.
- **Cost optimization** – The pay-per-token model in Amazon Bedrock and RAG architecture reduce inference and pre-training costs.
- **Sustainability** – Using serverless infrastructure and pay-per-token compute minimizes resource usage and enhances sustainability.

Use case: Predicting patient outcomes and re-admission rates

AI-powered predictive analytics offer further benefits by forecasting patient outcomes and enabling personalized treatment plans. This can improve patient satisfaction and health outcomes. By integrating these AI capabilities with Amazon Bedrock and other technologies, healthcare providers can achieve significant productivity gains, reduce costs, and elevate the overall quality of patient care.

You can store medical data, such as patient histories, clinical notes, medications, and treatments, in a knowledge graph. By combining the deep contextual understanding of LLMs with the structured, temporal data in a medical knowledge graph, healthcare providers can gain additional insights into individual patient patterns. Using predictive analytics, you can identify potential non-adherence or treatment complications early on and generate personalized re-admission propensity scores.

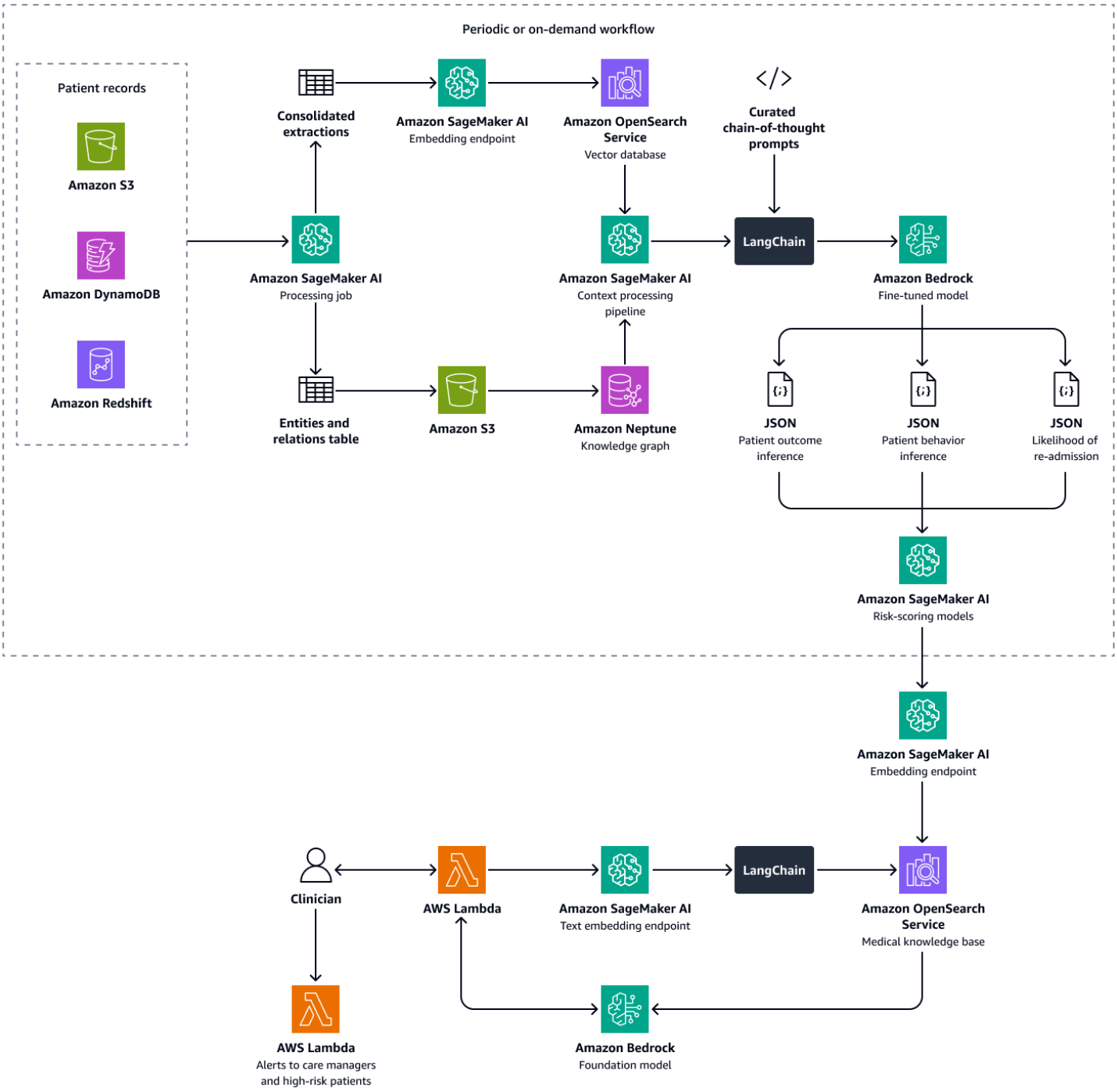
This solution helps you predict the likelihood of a re-admission. These predictions can improve patient outcomes and reduce healthcare costs. This solution can also help hospital clinicians and administrators focus their attention on patients with a higher risk of re-admission. It also helps them initiate proactive interventions with those patients through alerting, self-service, and data-driven actions.

Solution overview

This solution uses a multi-retriever Retrieval Augmented Generation (RAG) framework to analyze patient data. It predicts the likelihood of hospital re-admission for individual patients and helps you calculate a hospital-level re-admission propensity score. This solution integrates the following features:

- **Knowledge graph** – Stores structured, chronological patient data, such as hospital encounters, previous re-admissions, symptoms, lab results, prescribed treatments, and medication-adherence history
- **Vector database** – Stores unstructured clinical data, such as discharge summaries, physician notes, and records of missed appointments or reported medication side effects
- **Fine-tuned LLM** – Consumes both structured data from the knowledge graph and unstructured data from the vector database in order to generate inferences about a patient's behavior, treatment adherence, and re-admission likelihood

The risk-scoring models quantify the inferences from the LLM into numerical scores. You can aggregate the scores into a hospital-level re-admission propensity score. This score defines each patient's risk exposure, and you can calculate it periodically or on an as-needed basis. All inferences and risk scores are indexed and stored in Amazon OpenSearch Service so that care managers and clinicians can retrieve it. By integrating a conversational AI agent with this vector database, clinicians and care managers can seamlessly extract insights at an individual patient level, a facility-wide level, or by medical specialty. You can also set up automated alerts based on risk scores, which encourages proactive interventions.



Building this solution consists of the following steps:

- Step 1: Predicting patient outcomes by using a medical knowledge graph
- Step 2: Predicting patient behavior towards prescribed medications or treatments
- Step 3: Predicting patient re-admission likelihood

Step 1: Predicting patient outcomes by using a medical knowledge graph

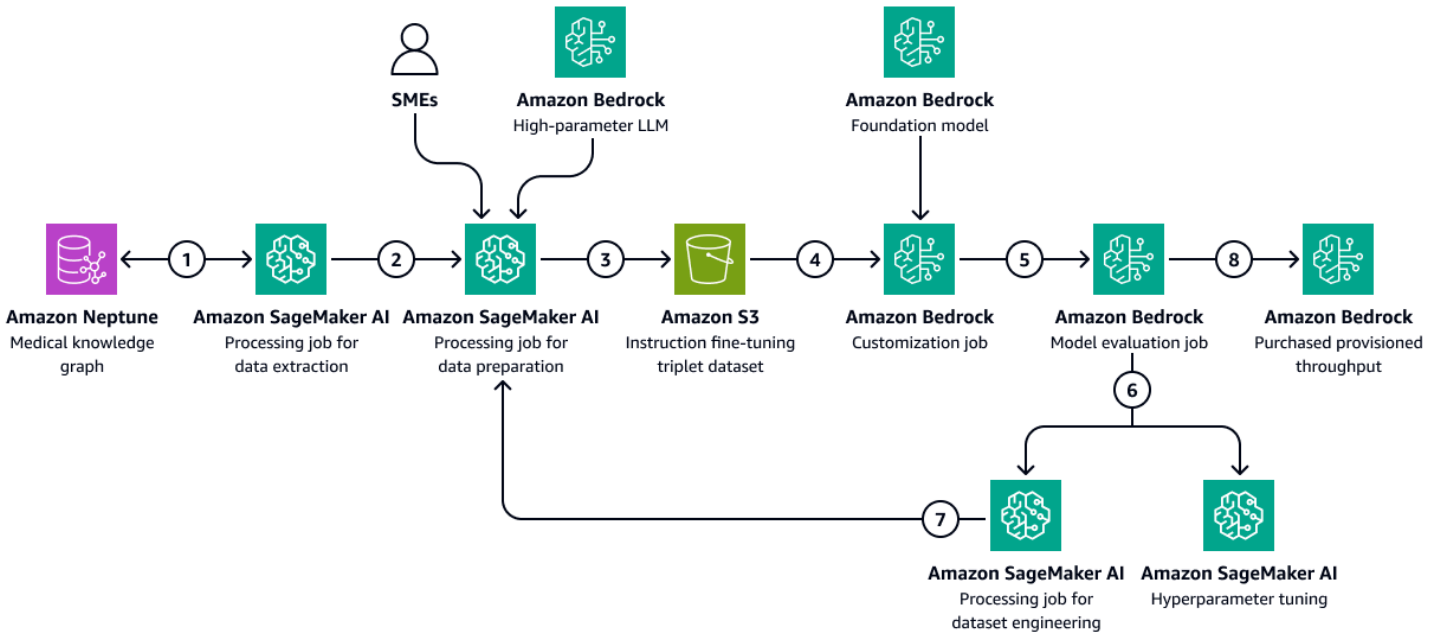
In Amazon Neptune, you can use a knowledge graph to store temporal knowledge about patient visits and outcomes over time. The most effective way to build and store a knowledge graph is to use a *graph model* and a *graph database*. Graph databases are purpose-built to store and navigate relationships. Graph databases make it easier to model and manage highly connected data and have flexible schemas.

The knowledge graph helps you perform time-series analysis. The following are key elements of the graph database that are used for temporal prediction of patient outcomes:

- **Historical data** – Prior diagnoses, continuing medications, medications previously used, and lab results for the patient
- **Patient visits (chronological)** – Visit dates, symptoms, observed allergies, clinical notes, diagnoses, procedures, treatments, prescribed medications, and lab results
- **Symptoms and clinical parameters** – Clinical and symptom-based information, including severity, patterns of progression, and the patient's response to medicine

You can use the insights from the medical knowledge graph to fine-tune an LLM in Amazon Bedrock, such as Llama 3. You fine-tune the LLM with sequential patient data about the patient's response to a set of medications or treatments over time. Use a labeled dataset that classifies a set of medications or treatments and patient-clinic interaction data in to predefined categories that indicate the health status of a patient. Examples of these categories are *deterioration in health*, *improvement*, or *stable progress*. When the clinician inputs new context about the patient and their symptoms, the fine-tuned LLM can use the patterns from the training dataset in order to predict the potential patient outcome.

The following image shows the sequential steps involved in fine-tuning an LLM in Amazon Bedrock by using a healthcare-specific training dataset. This data might include patient medical conditions and responses to treatments over time. This training dataset would help the model to make generalized predictions about patient outcomes.



The diagram shows the following workflow:

1. The Amazon SageMaker AI data extraction job queries the knowledge graph to retrieve chronological data about different patients' responses to a set of medications or treatments over time.
2. The SageMaker AI data preparation job integrates an Amazon Bedrock LLM and inputs from subject matter experts (SMEs). The job classifies the data retrieved from the knowledge graph into predefined categories (such as *deterioration in health*, *improvement*, or *stable progress*) that indicate the health status of each patient.
3. The job creates a fine-tuning dataset that includes the information extracted from the knowledge graph, the chain-of-thought prompts, and the patient outcome category. It uploads this training dataset to an Amazon S3 bucket.
4. An Amazon Bedrock customization job uses this training dataset to fine-tune an LLM.
5. The Amazon Bedrock customization job integrates the Amazon Bedrock foundational model of choice within the training environment. It starts the fine-tuning job and uses the training dataset and the training hyperparameters that you configure.
6. An Amazon Bedrock evaluation job evaluates the fine-tuned model by using a pre-designed model-evaluation framework.
7. If the model needs improvement, the training job runs again with more data after careful consideration of the training dataset. If the model does not demonstrate incremental performance improvement, also consider modifying the training hyperparameters.

8. After the model evaluation meets the standards defined by the business stakeholders, you host the fine-tuned model to the Amazon Bedrock provisioned throughput.

Step 2: Predicting patient behavior towards prescribed medications or treatments

Fine-tuned LLMs can process clinical notes, discharge summaries, and other patient-specific documents from the temporal medical knowledge graph. They can assess whether the patient is likely to follow prescribed medications or treatments.

This step uses the knowledge graph created in Step 1: Predicting patient outcomes by using a medical knowledge graph. The knowledge graph contains data from the patient's profile, including the patient's historical adherence as a node. It also includes instances of non-adherence to medicines or treatments, side effects to medicines, lack of access or cost barriers to medications, or complex dosing regimens as attributes of such nodes.

Fine-tuned LLMs can consume past prescription fulfillment data from the medical knowledge graph and descriptive summaries of the clinical notes from an Amazon OpenSearch Service vector database. These clinical notes might mention frequently missed appointments or non-compliance with treatments. The LLM can use these notes to predict the likelihood of future non-adherence.

1. Prepare the input data as follows:
   - **Structured data** – Extract recent patient data, such as the last three visits and the lab results, from the medical knowledge graph.
   - **Unstructured data** – Retrieve the recent clinical notes from the Amazon OpenSearch Service vector database.
2. Construct an input prompt that includes patient history and current context. The following is an example prompt:

```
You are a highly specialized AI model trained in healthcare predictive analytics.
 Your task is to analyze a patient's historical medical records, adherence patterns,
 and clinical context to predict the **likelihood of future non-adherence** to
 prescribed medications or treatments.

### **Patient Details**
- **Patient ID:** {patient_id}
- **Age:** {age}
```

```
- **Gender:** {gender}
- **Medical Conditions:** {medical_conditions}
- **Current Medications:** {current_medications}
- **Prescribed Treatments:** {prescribed_treatments}

### **Chronological Medical History**
- **Visit Dates & Symptoms:** {visit_dates_symptoms}
- **Diagnoses & Procedures:** {diagnoses_procedures}
- **Prescribed Medications & Treatments:** {medications_treatments}
- **Past Adherence Patterns:** {historical_adherence}
- **Instances of Non-Adherence:** {past_non_adherence}
- **Side Effects Experienced:** {side_effects}
- **Barriers to Adherence (e.g., Cost, Access, Dosing Complexity):** {barriers}

### **Patient-Specific Insights**
- **Clinical Notes & Discharge Summaries:** {clinical_notes}
- **Missed Appointments & Non-Compliance Patterns:** {missed_appointments}

### **Let's think Step-by-Step to predict the patient behaviour**
1. You should first analyze past adherence trends and patterns of non-adherence.
2. Identify potential barriers, such as financial constraints, medication side
 effects, or complex dosing regimens.
3. Thoroughly examine clinical notes and documented patient behaviors that may hint
 at non-adherence.
4. Correlate adherence history with prescribed treatments and patient conditions.
5. Finally predict the likelihood of non-adherence based on these contextual
 insights.

### **Output Format (JSON)**
Return the prediction in the following structured format:
```json
{
  "patient_id": "{patient_id}",
  "likelihood_of_non_adherence": "{low | moderate | high}",
  "reasoning": "{detailed_explanation_based_on_patient_history}"
}
```

3. Pass the prompt to the fine-tuned LLM. The LLM processes the prompt and predicts the outcome. The following is an example response from the LLM:

```
{
  "patient_id": "P12345",
  "likelihood_of_non_adherence": "high",
```
```

```
  "reasoning": "The patient has a history of missed appointments, has reported side
 effects to previous medications. Additionally, clinical notes indicate difficulty
 following complex dosing schedules."
}
```

4. Parse the model's response to extract the predicted outcome category. For example, the category for the example response in the previous step might be *high likelihood of non-adherence*.
5. (Optional) Use model logits or additional methods to assign confidence scores. *Logits* are the unnormalized probabilities of the item belonging to a certain class or category.

Step 3: Predicting patient re-admission likelihood

Hospital re-admissions are a major concern due to the high cost of healthcare administration and because of their impact on the patient's well-being. Computing hospital re-admission rates is one way of measuring the quality of patient care and performance of a healthcare provider.

To compute the re-admission rate, you defined an indicator, such as a 7-day re-admission rate. This indicator is the percentage of admitted patients who return to the hospital for an unplanned visit within seven days of discharge. To predict the chance of re-admission for a patient, a fine-tuned LLM can consume temporal data from the medical knowledge graph that you created in Step 1: Predicting patient outcomes by using a medical knowledge graph. This knowledge graph maintains chronological records of patient encounters, procedures, medications, and symptoms. These data records contain the following:

- Duration of time since the patient's last discharge
- The patient's response to past treatments and medications
- The progression of symptoms or conditions over time

You can process these time-series events to predict the re-admission likelihood of a patient through a curated system prompt. The prompt imparts the prediction logic to the fine-tuned LLM.

1. Prepare the input data as follows:
   - **Adherence history** – Extract medication pickup dates, medication refill frequencies, diagnosis and medication details, chronological medical history, and other information from the medical knowledge graph.

- **Behavioral indicators** – Retrieve and include clinical notes about missed appointments and patient-reported side effects.

2. Construct an input prompt that includes the adherence history and behavioral indicators. The following is an example prompt:

```
You are a highly specialized AI model trained in healthcare predictive analytics.
 Your task is to analyze a patient's historical medical records, clinical events, and
 adherence patterns to predict the **likelihood of hospital readmission** within the
 next few days.

### **Patient Details**
- **Patient ID:** {patient_id}
- **Age:** {age}
- **Gender:** {gender}
- **Primary Diagnoses:** {diagnoses}
- **Current Medications:** {current_medications}
- **Prescribed Treatments:** {prescribed_treatments}

### **Chronological Medical History**
- **Recent Hospital Encounters:** {encounters}
- **Time Since Last Discharge:** {time_since_last_discharge}
- **Previous Readmissions:** {past_readmissions}
- **Recent Lab Results & Vital Signs:** {recent_lab_results}
- **Procedures Performed:** {procedures_performed}
- **Prescribed Medications & Treatments:** {medications_treatments}
- **Past Adherence Patterns:** {historical_adherence}
- **Instances of Non-Adherence:** {past_non_adherence}

### **Patient-Specific Insights**
- **Clinical Notes & Discharge Summaries:** {clinical_notes}
- **Missed Appointments & Non-Compliance Patterns:** {missed_appointments}
- **Patient-Reported Side Effects & Complications:** {side_effects}

### **Reasoning Process – You have to analyze this use case step-by-step.**
1. First assess **time since last discharge** and whether recent hospital encounters
 suggest a pattern of frequent readmissions.
2. Second examine **recent lab results, vital signs, and procedures performed** to
 identify clinical deterioration.
3. Third analyze **adherence history**, checking if past non-adherence to medications
 or treatments correlates with readmissions.
4. Then identify **missed appointments, self-reported side effects, or symptoms
 worsening** from clinical notes.
```

```
5. Finally predict the **likelihood of readmission** based on these contextual
 insights.

### **Output Format (JSON)**
Return the prediction in the following structured format:
```json
{
  "patient_id": "{patient_id}",
  "likelihood_of_readmission": "{low | moderate | high}",
  "reasoning": "{detailed_explanation_based_on_patient_history}"
}
```

3. Pass the prompt to the fine-tuned LLM. The LLM processes the prompt and predicts the re-admission likelihood and reasons. The following is an example response from the LLM:

```
{
  "patient_id": "P67890",
  "likelihood_of_readmission": "high",
  "reasoning": "The patient was discharged only 5 days ago, has a history of more
 than two readmissions to hospitals where the patient received treatment. Recent
 lab results indicate abnormal kidney function and high liver enzymes. These factors
 suggest a medium risk of readmission."
}
```

4. Categorize the prediction into a standardized scale, such as *low*, *medium*, or *high*.
5. Review the reasoning provided by the LLM, and identify key factors that contribute to the prediction.
6. Map the qualitative outputs to quantitative scores. For example, *very high* might equal a 0.9 probability.
7. Use validation datasets to calibrate the model outputs against actual re-admission rates.

Step 4: Computing the hospital re-admission propensity score

Next, you calculate a hospital re-admission propensity score per patient. This score reflects the net impact of the three analyses performed in the previous steps: potential patient outcomes, patient behavior towards medications and treatments, and patient re-admission likelihood. By aggregating the patient-level re-admission propensity score to specialty level and then at hospital level, you can gain insights to clinicians, care managers, and administrators. The hospital re-admission propensity
```

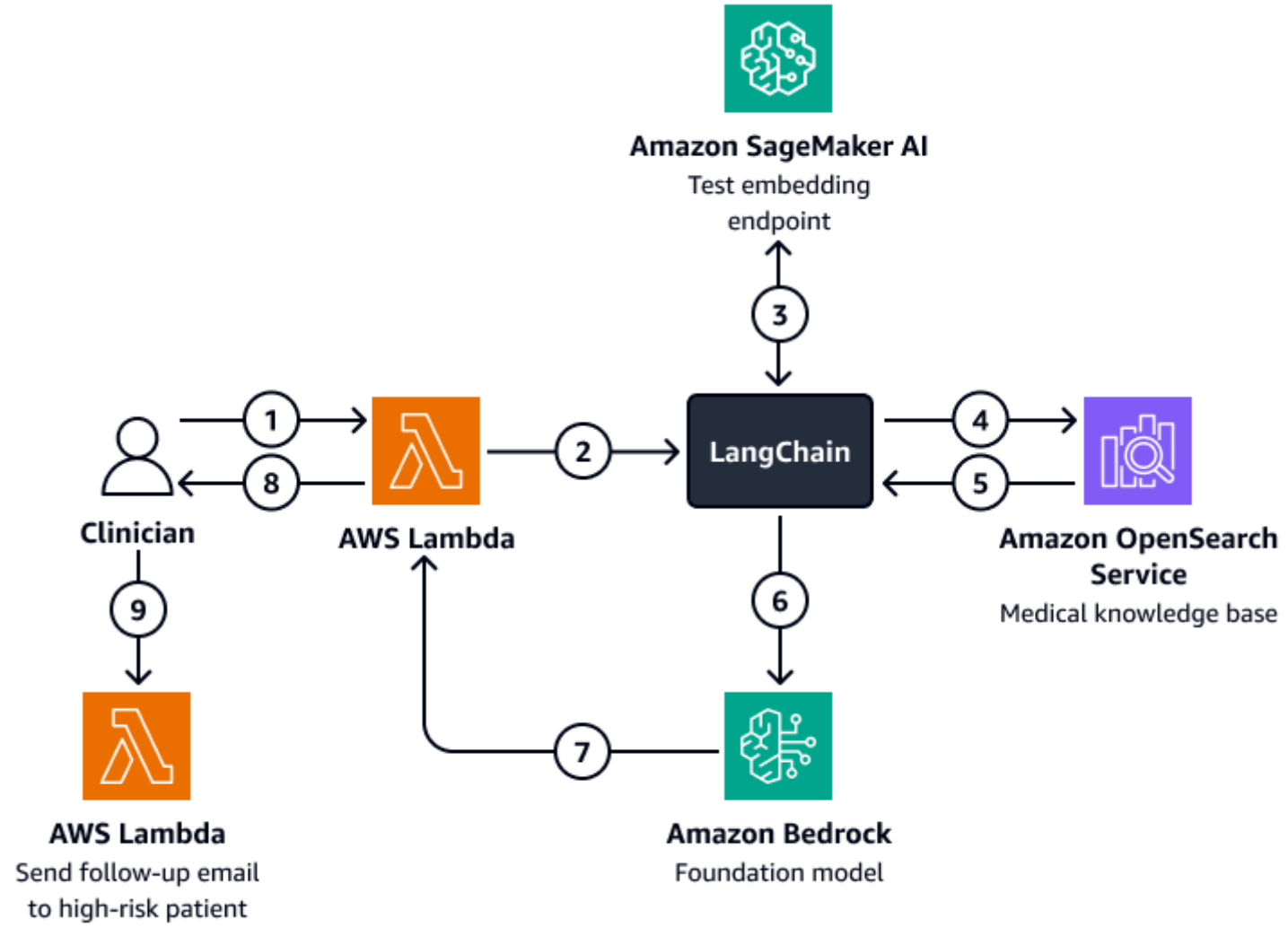
score helps you assess overall performance by facility, by specialty, or by condition. Then, you can use this score to implement proactive interventions.

1. Assign weights to each of the different factors (outcome prediction, adherence likelihood, re-admission). The following are example weights:
   - Outcome Prediction Weight: 0.4
   - Adherence Prediction Weight: 0.3
   - Re-admission Likelihood Weight: 0.3
2. **Use the following calculation to calculate the composite score:**

   `ReadadmissionPropensityScore` = (`OutcomeScore` × `OutcomeWeight`) + (`AdherenceScore` × `AdherenceWeight`) + (`ReadmissionLikelihoodScore` × `ReadmissionLikelihoodWeight`)
3. Make sure that all individual scores are on the same scale, such as 0 to 1.
4. Define the thresholds for action. For example, scores above 0.7 initiate alerts.

Based on the above analyses and re-admission propensity score of a patient, clinicians or care managers can set up alerts to monitor their individual patients based on the computed score. If it is above a pre-defined threshold, they are notified when that threshold is reached. This helps care managers to be proactive rather than reactive when creating discharge care plans for their patients. Save the patient outcome, behavior, and re-admission propensity scores in an indexed form in an Amazon OpenSearch Service vector database so that care managers can seamlessly retrieve them by using a conversational AI agent.

The following diagram shows the workflow of a conversational AI agent that a clinician or care manager can use to retrieve insights on patient outcomes, expected behavior, and re-admission propensity. Users can retrieve insights at the patient-level, department-level, or hospital-level. The AI agent retrieves these insights, which are stored in an indexed form in an Amazon OpenSearch Service vector database. The agent uses the query to retrieve relevant data and provides tailored responses, including suggested actions for patients who have a high risk of re-admission. Based on the level of risk, the agent can also set up reminders for patients and care givers.



The diagram shows the following workflow:

1. The clinician poses a question to a conversational AI agent, which houses an AWS Lambda function.
2. The Lambda function initiates a LangChain agent.
3. The LangChain agent sends the user's question to an Amazon SageMaker AI text embedding endpoint. The endpoint embeds the question.
4. The LangChain agent passes embedded question to a medical knowledge base in Amazon OpenSearch Service.
5. Amazon OpenSearch Service returns the specific insights that are most relevant to the user query to the LangChain agent.
6. The LangChain agents sends the query and the retrieved context from the knowledge base to an Amazon Bedrock foundation model.

7. The Amazon Bedrock foundation model generates a response and sends it to the Lambda function.
8. The Lambda function returns the response to the clinician.
9. The clinician initiates a Lambda function that sends a follow-email to a patient who has a high risk of re-admission.

Alignment to the AWS Well-Architected Framework

The architecture for tracking patient behavior and predicting hospital re-admission rates integrates AWS services, medical knowledge graphs, and LLMs to improve healthcare outcomes while aligning with the six pillars of the AWS Well-Architected Framework:

- **Operational excellence** – The solution is a decoupled, automated system that uses Amazon Bedrock and AWS Lambda for real-time alerts.
- **Security** – This solution is designed to comply with healthcare regulations, such as HIPAA. You can also implement encryption, fine-grained access control, and Amazon Bedrock guardrails to help protect patient data.
- **Reliability** – The architecture uses fault-tolerant, serverless AWS services.
- **Performance efficiency** – Amazon OpenSearch Service and the fine-tuned LLMs can provide fast and accurate predictions.
- **Cost optimization** – Serverless technologies and pay-per-inference models help minimize costs. Although a using fine-tuned LLM can incur extra charges, the model uses a RAG approach that reduces the data and computational time required for the fine-tuning process.
- **Sustainability** – The architecture minimizes resource consumption through the use of serverless infrastructure. It also supports efficient, scalable healthcare operations.

Use case: Managing and upskilling your healthcare staff

Implementing talent-transformation and upskilling strategies helps workforces remain adept at using new technologies and practices in medical and healthcare services. Proactive upskilling initiatives make sure that healthcare professionals can provide high-quality patient care, optimize operational efficiencies, and stay compliant with regulatory standards. Moreover, talent transformation fosters a culture of continuous learning. This is pivotal for adapting to the changing healthcare landscape and addressing emerging public health challenges. Traditional training approaches, such as classroom-based training and static learning modules, offer uniform content to a broad audience. They often lack personalized learning paths, which are critical for addressing the specific needs and proficiency levels of individual practitioners. This one-size-fits-all strategy can result in disengagement and suboptimal knowledge retention.

Consequently, healthcare organizations must embrace innovative, scalable, and technology-driven solutions that can determine the gap for each of their employees in their current state and potential future state. These solutions should recommend hyper-personalized learning pathways and the right set of learning content. This effectively prepares the workforce for the future of healthcare.

In the healthcare industry, you can apply generative AI to help you understand and upskill your workforce. Through the connection of large language models (LLMs) and advanced retrievers, organizations can understand what skills they currently have and identify key skills that might be necessary in the future. This information helps you bridge the gap by hiring new workers and upskilling the current workforce. Using Amazon Bedrock and knowledge graphs, healthcare organizations can develop domain-specific applications that facilitate continuous learning and skill development.

The knowledge provided by this solution helps you effectively manage talent, optimize workforce performance, drive organizational success, identify existing skills, and craft a talent strategy. This solution can help you perform these tasks in weeks instead of months.

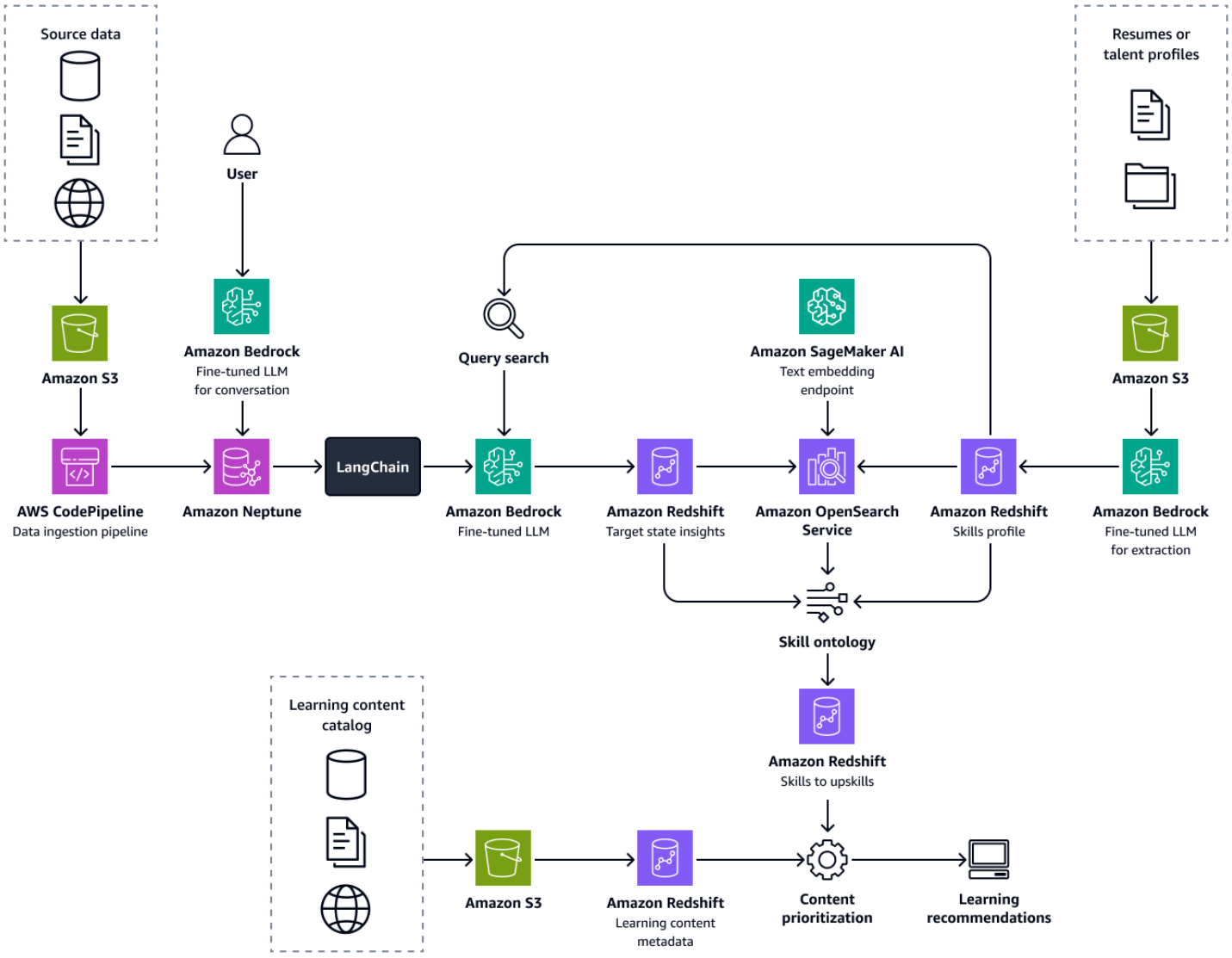
Solution overview

This solution is a healthcare talent transformation framework that consists of the following components:

- **Intelligent resume parser** – This component can read a candidate's resume and precisely extract candidate information, including skills. Intelligent information extraction solution built using fine-tuned Llama 2 model in Amazon Bedrock on a proprietary training dataset covering resumes and talent profiles from more than 19 industries. This LLM-based process saves hundreds of hours by automating the manual review process of resumes and matching top candidates to open roles.
- **Knowledge graph** – A knowledge graph built on Amazon Neptune, a unified repository of talent information including role and skill taxonomy of the organization as well as the industry, capturing the semantics of healthcare talent using definitions of skills, roles and their properties, relations, and logical constraints.
- **Skill ontology** – The discovery of skill proximities between candidate skills and ideal current state or future state skills (retrieved using a knowledge graph) is achieved through ontology algorithms that measure the semantic similarity between candidate skills and target state skills.
- **Learning pathway and content** – This component is a learning recommendation engine that can recommend the right learning content from a catalogue of learning materials from any vendor based on the identified skill gaps. Identifying the most optimal upskilling pathways for each candidate by analyzing the skill gaps and recommending prioritized learning content, to enable a seamless and continuous professional development for each candidate during the transition to a new role.

This cloud-based, automated solution is powered by machine learning services, LLMs, knowledge graphs, and Retrieval Augmented Generation (RAG). It can scale to process tens or thousands of resumes in minimum amount of time, create instant candidate profiles, identify gaps in their current or potential future state, and then efficiently recommend the right learning content to close these gaps.

The following image shows the end-to-end flow of the framework. The solution is built on fine-tuned LLMs in Amazon Bedrock. These LLMs retrieve data from the healthcare talent knowledge base in Amazon Neptune. Data-driven algorithms make recommendations for optimal learning pathways for each candidate.



Building this solution consists of the following steps:

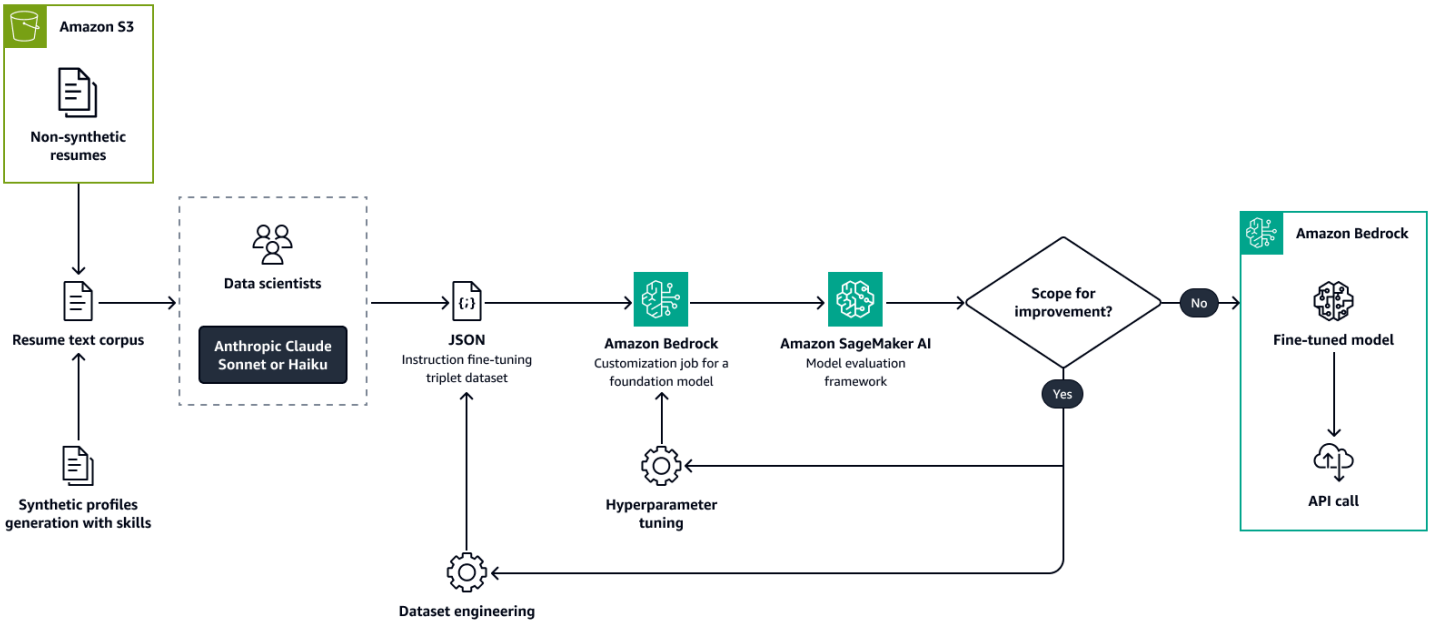
- Step 1: Extracting talent information and building a skills profile
- Step 2: Discovering role-to-skill relevance from a knowledge graph
- Step 3: Identifying skill gaps and recommending training

Step 1: Extracting talent information and building a skills profile

First, you fine-tune a large language model, such as Llama 2, in Amazon Bedrock with a custom dataset. This adapts the LLM for the use case. During training, you accurately and consistently

extract key talent attributes from candidate resumes or similar talent profiles. These talent attributes include skills, current role title, experience titles with date spans, education, and certifications. For more information, see Customize your model to improve its performance for your use case in the Amazon Bedrock documentation.

The following image shows the process to fine-tune a resume-parsing model by using Amazon Bedrock. Both real and synthetically created resumes are passed to an LLM in order to extract key information. A group of data scientists validate the extracted information against the original, raw text. The extracted information is then concatenated by using chain-of-thought prompting and the original text to derive a training dataset for fine-tuning. This dataset is then passed to an Amazon Bedrock customization job, which fine-tunes the model. An Amazon SageMaker AI batch job runs a model-evaluation framework that evaluates the fine-tuned model. If the model needs improvement, the job runs again with more data or different hyperparameters. After the evaluation meets the standards, you host the custom Model through Amazon Bedrock provisioned throughput.



Step 2: Discovering role-to-skill relevance from a knowledge graph

Next, you build a knowledge graph that encapsulates the skills and role taxonomy of your organization and of other organizations in the healthcare industry. This enriched knowledge base is sourced from aggregated talent and organization data in Amazon Redshift. You can gather talent data from a range of labor-market data providers and from organization-specific structured and

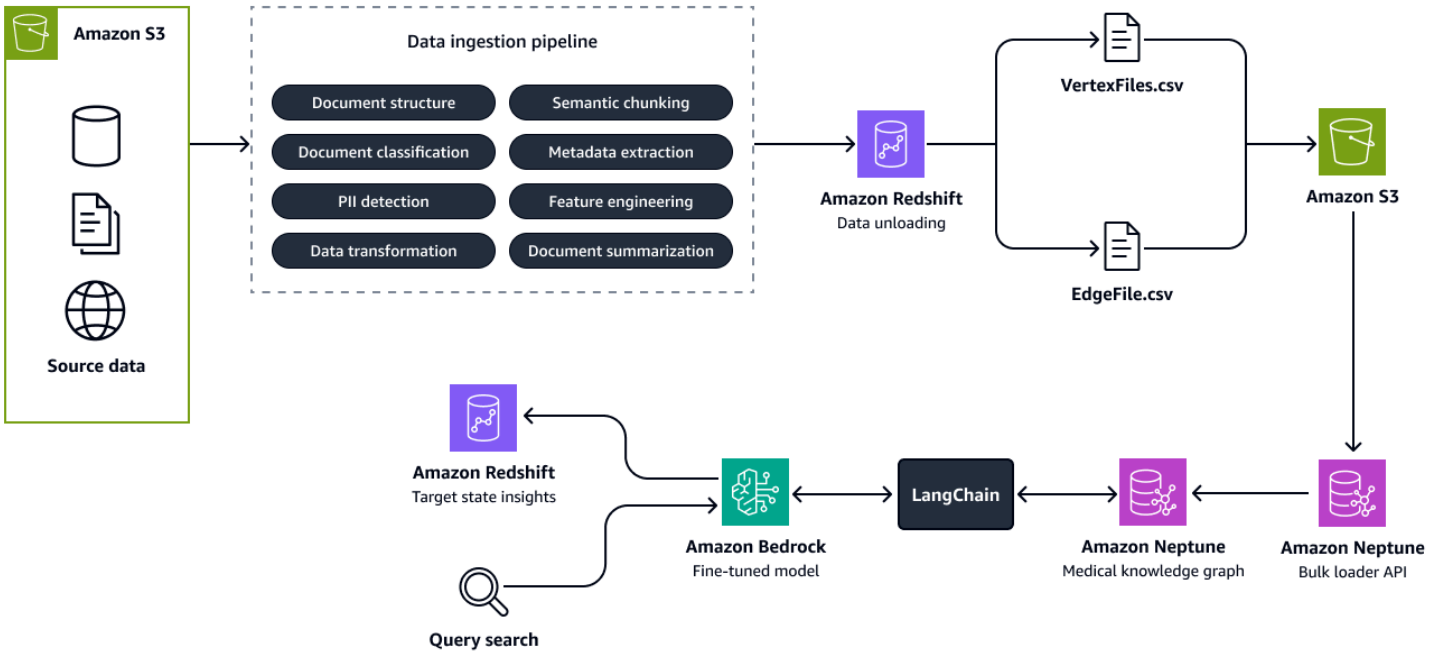
unstructured data sources, such as enterprise resource planning (ERP) systems, a human resources information system (HRIS), employee resumes, job descriptions, and talent architecture documents.

Build the knowledge graph on Amazon Neptune. Nodes represent skills and roles, and edges represent the relationships between them. Enrich this graph with metadata to include details such as organization name, industry, job family, skill type, role type, and industry tags.

Next, you develop a *Graph Retrieval Augmented Generation (Graph RAG)* application. Graph RAG is a RAG approach that retrieves data from a graph database. The following are the components of the Graph RAG application:

- **Integration with an LLM in Amazon Bedrock** – The application uses an LLM in Amazon Bedrock for natural language understanding and query generation. Users can interact with the system by using natural language. This makes it accessible to non-technical stakeholders.
- **Orchestration and information retrieval** – Use LlamaIndex or LangChain orchestrators to facilitate the integration between the LLM and the Neptune knowledge graph. They manage the process of converting natural language queries into openCypher queries. Then, they run the queries on the knowledge graph. Use prompt engineering to instruct the LLM about best practices for constructing openCypher queries. This helps optimize the queries to retrieve the relevant subgraph, which contains all of the pertinent entities and relationships about the queried roles and skills.
- **Insight generation** – The LLM in Amazon Bedrock processes the retrieved graph data. It generates detailed insights about the current state and projects future states for the queried role and associated skills.

The following image shows the steps to build a knowledge graph from source data. You pass the structured and unstructured source data to the data ingestion pipeline. The pipeline extracts and transforms information to a CSV bulk load formation that is compatible with Amazon Neptune. The bulk loader API uploads the CSV files that are stored in an Amazon S3 bucket to the Neptune knowledge graph. For user queries related to talent future state, relevant roles, or skills, the fine-tuned LLM in Amazon Bedrock interacts with the knowledge graph through a LangChain orchestrator. The orchestrator retrieves the relevant context from the knowledge graph and push the responses to the insights table in Amazon Redshift. The LangChain orchestrator, like GraphQAChain, converts the natural language query from the user to an openCypher query in order to query the knowledge graph. The Amazon Bedrock fine-tuned model generates a response based on the retrieved context.



Step 3: Identifying skill gaps and recommending training

In this step, you accurately compute the proximity between the current state of a healthcare professional and potential future state roles. To do this, you perform a skill affinity analysis by comparing the individual's skills sets with the job role. In an Amazon OpenSearch Service vector database, you store skill taxonomy information and skill metadata, such as the skill description, skill type, and skill clusters. Use an Amazon Bedrock embedding model, such as Amazon Titan Text Embeddings models, to embed the identified key skill into vectors. Through a vector search, you retrieve the descriptions of current state skills and target state skills and perform an ontology analysis. The analysis provides proximity scores between the current and target state skill pairs. For each pair, you use the computed ontology scores to identify the gaps in skill affinities. Then, you recommend the optimal path for upskilling, which the candidate can consider during role transitions.

For each role, recommending the correct learning content for upskilling or reskilling involves a systematic approach that begins with creating a comprehensive catalog of learning content. This catalog, which you store in an Amazon Redshift database, aggregates content from various providers and includes metadata, such as the content duration, difficulty level, and learning mode. The next step is to extract the key skills offered by each piece of content and then map them to the individual skills required for the target role. You achieve this mapping by analyzing the coverage provided by the content through a skills proximity analysis. This analysis assesses how closely

the skills taught by the content align with the desired skills for the role. The metadata plays a critical role in selecting the most appropriate content for each skill, making sure that learners receive tailored recommendations that suit their learning needs. Use LLMs in Amazon Bedrock to extract skills from the content metadata, perform feature engineering, and validate the content recommendations. This improves accuracy and relevance in the upskilling or reskilling process.

Alignment to the AWS Well-Architected Framework

The solution aligns with all six pillars of the AWS Well-Architected Framework:

- **Operational excellence** – A modular, automated pipeline enhances operational excellence. Key components of the pipeline are decoupled and automated, allowing for faster model updates and easier monitoring. Additionally, automated training pipelines support quicker releases of fine-tuned models.
- **Security** – This solution processes sensitive and personally identifiable information (PII), such as the data in resumes and talent profiles. In AWS Identity and Access Management (IAM), implement fine-grained access control policies and make sure that only authorized personnel have access to this data.
- **Reliability** – The solution uses AWS services, such as Neptune, Amazon Bedrock, and OpenSearch Service, that provide fault tolerance, high availability, and uninterrupted access to insights even during high demand.
- **Performance efficiency** – Fine-tuned LLMs in Amazon Bedrock and OpenSearch Service vector databases are designed to quickly and accurately process large datasets in order to deliver timely, personalized learning recommendations.
- **Cost optimization** – This solution uses a RAG approach, which reduces the need for continuous pre-training of models. Instead of fine-tuning the entire model repeatedly, the system fine-tunes only specific processes, such as extracting information from resumes and structuring outputs. This results in significant cost savings. By minimizing the frequency and scale of resource-intensive model training and by using pay-per-use cloud services, healthcare organizations can optimize their operational costs while maintaining high performance.
- **Sustainability** – This solution uses scalable, cloud-native services that allocate compute resources dynamically. This reduces energy consumption and the environmental impact while still supporting large-scale, data-intensive talent transformation initiatives.

Developing and orchestrating generative AI solutions for healthcare

To build the solutions in this guide, you must build a RAG architecture that uses fine-tuned LLMs to deliver augmented patient data, clinical and diagnostic insights, and predicted patient outcomes to healthcare providers. This requires the integration of multiple AWS services and tools to create a cohesive and efficient workflow. This section discusses the following:

- Amazon Q Developer – Use Amazon Q Developer to address engineering questions and code errors during the development process.
- Multi-retriever RAG design – Design and implement RAG solutions that use multiple retrievers to fetch the correct medical context for the user's question.
- ReAct agents – Implement agents that combine reasoning with dynamic action.

Amazon Q Developer

When building a generative AI solution, it can be difficult to create AI agents and the connect key services. However, Amazon Q Developer helps data scientists and AI engineers by providing access to an advanced generative AI assistant. Amazon Q can quickly and accurately address user questions and code errors, which can help you optimize the LLM development process. Amazon Q offers significant advantages for developers creating applications that use Amazon Bedrock foundation models. It can streamline workflows and enhance code quality. It automates the generation of Python scripts and infrastructure as code (IaC) configurations, significantly reducing development time and effort. Through advanced refactoring capabilities, Amazon Q can improve code performance, identify security vulnerabilities, and make sure developers adhere to best practices. Additionally, it facilitates learning and adoption for beginners by providing context-aware suggestions and explanations, making complex coding tasks more accessible and efficient.

Multi-retriever RAG design

In a generative AI application, a multi-retriever RAG pipeline can efficiently retrieve information from multiple data sources to help healthcare providers and clinicians answer medical questions. This pipeline uses different types of retrievers to pull relevant data from different knowledge

bases. Each retriever is specialized in fetching a particular type of information, such as patient histories, diagnostic insights, clinical notes, or content from medical research and academic texts.

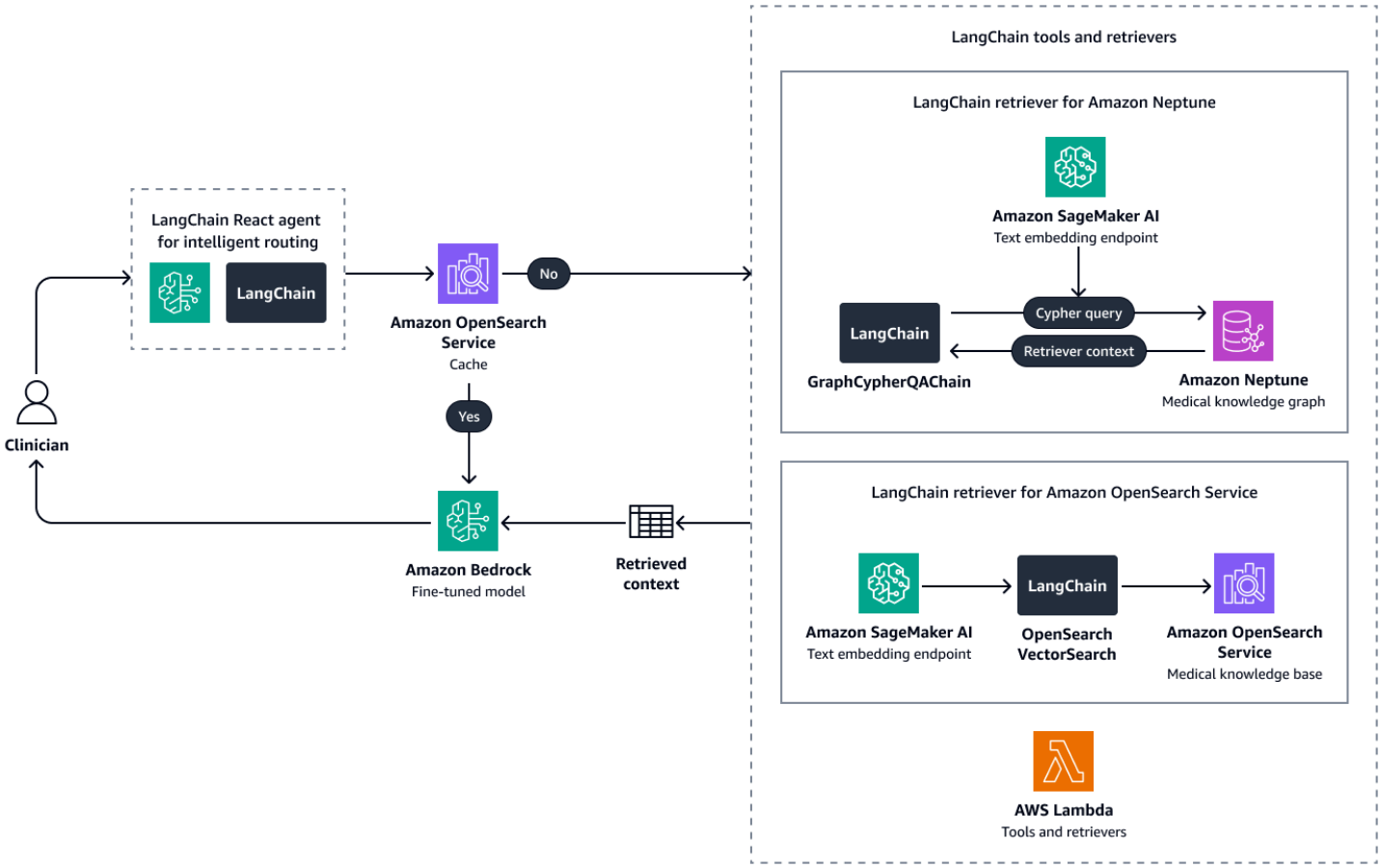
Use the nature of the data and the specific application requirements to determine what the correct backend knowledge base is correct for your use case. An Amazon OpenSearch Service vector database is well suited for large volumes of unstructured or semi-structured healthcare data, including image diagnosis assessment summaries, discharge summaries, clinical reports, medical research, and academic text content. On the other hand, a graph database service, such as Amazon Neptune, can be ideal for healthcare use cases that require deep exploration of the temporal relationships between entities, such as patient, patient history, healthcare provider, medicines, symptoms, and treatments.

A critical component of this pipeline is *user query intent prediction*. This makes sure that the system routes the query to the correct retriever chain. For example, if a clinician asks about a patient's treatment history, symptoms, interaction with the hospital, the likelihood of hospital re-admission, or potential patient outcomes, then the query intent prediction module identifies this intent. It directs the request to the retriever chain that can fetch patient records or chronological treatment data from the medical knowledge graph. Alternatively, if the question is about disease discovery, specific diagnostic assessments, or details of specific clinical procedures from academic textbooks, then the query is routed to the retriever chain that can fetch this information from the OpenSearch Service vector database. You can use the tool-calling functionality of LangChain to bind a custom tool to the Amazon Bedrock LLM that can classify a user question into predefined intents.

This multi-retriever RAG system includes LangChain agents that are designed to manage access to the specific knowledge base. You can use LangChain to orchestrate the interaction between the Amazon Bedrock LLM, the different retrievers, and tools. LangChain includes a tool-calling class that helps you create custom tools, such as an intent classifier, a retriever for Neptune, a retriever for OpenSearch Service, or any other tool that can be developed to classify the user intent and access data from a specific knowledge base in a structured format. You then feed these tools to the class to create a Reasoning and Acting (ReAct) agent. The ReAct agent processes the user question, plans the sequential steps to answer the question, and then iteratively executes the available tools and processes the tool responses to finally answer the user query.

The following image shows how a multi-retriever RAG system that is designed for efficient knowledge retrieval and intelligent query resolution works. A LangChain ReAct agent analyzes the user's intent, formulates a structured plan for execution, and selects the most relevant retrieval tools. The system queries a previous questions cache and checks for similar queries based on key attributes, such as patient ID, medical condition, and visit date. If a highly similar

question is found, the corresponding answer is retrieved directly. Otherwise, the agent executes the appropriate retriever. For retrieving patient-centric information, such as treatment history, symptoms, hospital interactions, or re-admission likelihood, the system uses a graph retriever. For diagnostic assessments, clinical procedures, and structured medical findings, the agent employs a vector database retriever. In scenarios that require a combination of contextual knowledge from both data stores, to generate a comprehensive response, the system uses a hybrid retrieval strategy that integrates results from both the knowledge graph and vector database.



ReAct agents

Reasoning and Acting (ReAct) agents are designed for multi-faceted RAG applications. These agents provide a powerful combination of reasoning and dynamic action, particularly for complex applications that involve step-by-step, logical information-retrieval workflows. For more information, see ReAct: Synergizing Reasoning and Acting in Language Models.

In medical and healthcare contexts, queries from a clinician or doctor are often multi-faceted. For example, a clinician might ask "What treatments were given to similar patients with both

hypertension and type 2 diabetes?" After identifying the user intent, which is to fetch the treatments for hypertension and type 2 diabetes, the AI agent needs to divide this query into subtasks and then choose the most efficient retrieval strategy. In this case, the AI agent should identify the most relevant nodes (such as patient age, sex, conditions, treatments, and medications) and then query the graph for these entities and their attributes and relationships. ReAct agents are very helpful because they combine the reasoning (logical inference) capability of an LLM with an action (querying or interacting with external resources or knowledge bases).

To answer the user query "What treatments were given to similar patients with both hypertension and type 2 diabetes?", the following example illustrates how a ReAct agent works:

1. **Agent reasoning** – The ReAct agent infers that the question involves retrieving information about conditions (diabetes and hypertension). It considers patient age, treatments, medications, and the period to analyze.
2. **Agent action** – The agent uses openCypher to query the knowledge graph for treatments that are specific to type 2 diabetes and hypertension. It also retrieves medications administered, dates of hospital visits, side effects of medications, known patient outcomes, and cross-reference data for similar patients (such as patients of the same gender and age).
3. **Agent observation** – From the knowledge graph, the agent retrieves the most recent six months of tabular data about treatments given to patients who have both hypertension and type 2 diabetes.
4. **Agent reasoning** – To rank the results from the retrieved records, the agent identifies important attributes, such as recency, side effects of medications, or known patient outcomes.
5. **Agent action** – The agent re-ranks the records based on identified attributes and pre-defined logic that is imparted through the system prompt.
6. **Response generation** – The LLM in Amazon Bedrock generates a response based on the context that the ReAct agent prepared.

Evaluating generative AI solutions for healthcare

Evaluating the healthcare AI solutions you build is critical to making sure that they are effective, reliable, and scalable in real-world medical environments. Use a systematic approach to evaluate the performance of each component of the solution. The following is a summary of the methodologies and metrics that you can use to evaluate your solution.

Topics

Evaluating the extraction of information

Evaluate the performance of information extraction solutions, such as the intelligent resume parser and the custom entity extractor. You can measure the alignment of these solutions' responses by using a test dataset. If you don't have a dataset that covers versatile healthcare talent profiles and patient medical records, you can create a custom test dataset by using the reasoning capability of an LLM. For example, you could use a large parameter model, such as Anthropic Claude models, to generate a test dataset.

The following are three key metrics that you can use for evaluating the information extraction models:

- **Accuracy and completeness** – These metrics evaluate the extent to which the output captured the correct and complete information present in the ground truth data. This involves checking both the correctness of the extracted information and the presence of all relevant details in the extracted information.
- **Similarity and relevance** – These metrics assess the semantic, structural, and contextual similarities between the output and the ground truth data (the *similarity*) and the degree to which the output aligns with and addresses the content, context, and intent of the ground truth data (the *relevance*).
- **Adjusted recall or capture rate** – These rates empirically determine how many of the present values in the ground truth data were identified correctly by the model. The rate should include a penalization for all false values that the model extracts.

- **Precision score** – The precision score helps you determine how many false positives are present in the predictions, as compared to the true positives. For example, you can use precision metrics to measure the correctness of the extracted skill proficiency.

Evaluating RAG solutions with multiple retrievers

To assess how well the system retrieves relevant information and how effectively it uses that information to generate accurate and contextually appropriate responses, you can use the following metrics:

- **Response relevancy** – Measure how relevant the generated response, which uses the retrieved context, is to the original query.
- **Context precision** – Out of the total retrieved results, evaluate the proportion of retrieved documents or snippets that are relevant to the query. A higher context precision indicates that the retrieval mechanism is effective in selecting relevant information.
- **Faithfulness** – Assesses how accurately the generated response reflects the information in the retrieved context. In other words, measure if the response remains true to the source information.

Evaluating a solution by using an LLM

You can use a technique called *LLM-as-a-judge* to evaluate the text responses from your generative AI solution. It involves using LLMs to evaluate and assess the performance of model outputs. This technique uses the capabilities of Amazon Bedrock to provide judgments on various attributes, such as response quality, coherence, adherence, accuracy, and completeness to human preferences or ground truth data. You use chain-of-thought (CoT) and few-shot prompting techniques for a comprehensive evaluation. The prompt instructs the LLM to evaluate the generated response with scoring rubric, and the few-shot samples in the prompt demonstrate the actual evaluation process. The prompt also includes guidelines for the LLM evaluator to follow. For example, you could consider using one or more of the following evaluation techniques that use an LLM to judge the generated responses:

- **Pairwise comparison** – Give the LLM evaluator a medical question and multiple responses that were generated by different, iterative versions of the RAG systems you created. Prompt the LLM evaluator to determine the best response based on response quality, coherence, and adherence to the original question.

- **Single-answer grading** – This technique is well suited for use cases where you need to evaluate the accuracy of categorization, such as patient outcome classification, patient behavior categorization, patient re-admission likelihood, and risk categorization. Use the LLM evaluator to analyze individual categorization or classification in isolation, and evaluate the reasoning it has provided against ground truth data.
- **Reference-guided grading** – Provide the LLM evaluator with a series of medical questions that require descriptive answers. Create sample responses to these questions, such as reference answers or ideal responses. Prompt the LLM evaluator to compare the LLM-generated response against the reference answers or ideal responses, and prompt the LLM evaluator to grade the generated response for accuracy, completeness, similarity, relevance, or other attributes. This technique helps you evaluate whether the generated responses align with a well-defined standard or exemplary answer.

Resources

AWS documentation

- Amazon Bedrock documentation
- Amazon Neptune documentation
- Amazon OpenSearch Service documentation
- Applying the AWS Well-Architected Framework for Amazon Neptune (AWS Prescriptive Guidance)
- Operational best practices for Amazon OpenSearch Service (OpenSearch Service documentation)
- Using Amazon Comprehend Medical and LLMs for healthcare and life sciences (AWS Prescriptive Guidance)

AWS blog posts

- Build RAG and agent-based generative AI applications with new Amazon Titan Text Premier model, available in Amazon Bedrock
- Complement Commercial Intelligence by Building a Knowledge Graph out of a Data Warehouse with Amazon Neptune
- Using knowledge graphs to build GraphRAG applications with Amazon Bedrock and Amazon Neptune

Other resources

- Integrating Retrieval-Augmented Generation with Large Language Models in Nephrology: Advancing Practical Applications (PubMed Central, National Library of Medicine)
- Introduction to LangChain (LangChain documentation)

Contributors

Authoring

- Nitu Nivedita, Managing Director – Artificial Intelligence Lead, Data & AI, Accenture
- Manoj Appully, Founder and CTO, Cadiem
- Conor Folan, Consultant – Data & AI, Accenture
- Deepak Krishna AR, Consultant – Data & AI, Accenture
- Almore Cato, Manager – Data & AI, Accenture
- Soonam Kurian, Principal Solutions Architect, AWS

Reviewing

- Sally Lin, Data Science Senior Manager – Data & AI, Accenture
- Terry Huang, Data Science Manager – Data & AI, Accenture
- William Lorenz, Partners Solutions Architect, AWS

Technical writing

- Lilly AbouHarb, Senior Technical Writer, AWS

Document history

The following table describes significant changes to this guide. If you want to be notified about future updates, you can subscribe to an RSS feed.

Change	Description	Date
Initial publication	—	March 14, 2025

AWS Prescriptive Guidance glossary

The following are commonly used terms in strategies, guides, and patterns provided by AWS Prescriptive Guidance. To suggest entries, please use the **Provide feedback** link at the end of the glossary.

Numbers

7 Rs

Seven common migration strategies for moving applications to the cloud. These strategies build upon the 5 Rs that Gartner identified in 2011 and consist of the following:

- Refactor/re-architect – Move an application and modify its architecture by taking full advantage of cloud-native features to improve agility, performance, and scalability. This typically involves porting the operating system and database. Example: Migrate your on-premises Oracle database to the Amazon Aurora PostgreSQL-Compatible Edition.
- Replatform (lift and reshape) – Move an application to the cloud, and introduce some level of optimization to take advantage of cloud capabilities. Example: Migrate your on-premises Oracle database to Amazon Relational Database Service (Amazon RDS) for Oracle in the AWS Cloud.
- Repurchase (drop and shop) – Switch to a different product, typically by moving from a traditional license to a SaaS model. Example: Migrate your customer relationship management (CRM) system to Salesforce.com.
- Rehost (lift and shift) – Move an application to the cloud without making any changes to take advantage of cloud capabilities. Example: Migrate your on-premises Oracle database to Oracle on an EC2 instance in the AWS Cloud.
- Relocate (hypervisor-level lift and shift) – Move infrastructure to the cloud without purchasing new hardware, rewriting applications, or modifying your existing operations. You migrate servers from an on-premises platform to a cloud service for the same platform. Example: Migrate a Microsoft Hyper-V application to AWS.
- Retain (revisit) – Keep applications in your source environment. These might include applications that require major refactoring, and you want to postpone that work until a later time, and legacy applications that you want to retain, because there's no business justification for migrating them.

- Retire – Decommission or remove applications that are no longer needed in your source environment.

A

ABAC

See attribute-based access control.

abstracted services

See managed services.

ACID

See atomicity, consistency, isolation, durability.

active-active migration

A database migration method in which the source and target databases are kept in sync (by using a bidirectional replication tool or dual write operations), and both databases handle transactions from connecting applications during migration. This method supports migration in small, controlled batches instead of requiring a one-time cutover. It's more flexible but requires more work than active-passive migration.

active-passive migration

A database migration method in which the source and target databases are kept in sync, but only the source database handles transactions from connecting applications while data is replicated to the target database. The target database doesn't accept any transactions during migration.

aggregate function

A SQL function that operates on a group of rows and calculates a single return value for the group. Examples of aggregate functions include `SUM` and `MAX`.

AI

See artificial intelligence.

AIOps

See artificial intelligence operations.

anonymization

The process of permanently deleting personal information in a dataset. Anonymization can help protect personal privacy. Anonymized data is no longer considered to be personal data.

anti-pattern

A frequently used solution for a recurring issue where the solution is counter-productive, ineffective, or less effective than an alternative.

application control

A security approach that allows the use of only approved applications in order to help protect a system from malware.

application portfolio

A collection of detailed information about each application used by an organization, including the cost to build and maintain the application, and its business value. This information is key to the portfolio discovery and analysis process and helps identify and prioritize the applications to be migrated, modernized, and optimized.

artificial intelligence (AI)

The field of computer science that is dedicated to using computing technologies to perform cognitive functions that are typically associated with humans, such as learning, solving problems, and recognizing patterns. For more information, see What is Artificial Intelligence?

artificial intelligence operations (AIOps)

The process of using machine learning techniques to solve operational problems, reduce operational incidents and human intervention, and increase service quality. For more information about how AIOps is used in the AWS migration strategy, see the operations integration guide.

asymmetric encryption

An encryption algorithm that uses a pair of keys, a public key for encryption and a private key for decryption. You can share the public key because it isn't used for decryption, but access to the private key should be highly restricted.

atomicity, consistency, isolation, durability (ACID)

A set of software properties that guarantee the data validity and operational reliability of a database, even in the case of errors, power failures, or other problems.

attribute-based access control (ABAC)

The practice of creating fine-grained permissions based on user attributes, such as department, job role, and team name. For more information, see ABAC for AWS in the AWS Identity and Access Management (IAM) documentation.

authoritative data source

A location where you store the primary version of data, which is considered to be the most reliable source of information. You can copy data from the authoritative data source to other locations for the purposes of processing or modifying the data, such as anonymizing, redacting, or pseudonymizing it.

Availability Zone

A distinct location within an AWS Region that is insulated from failures in other Availability Zones and provides inexpensive, low-latency network connectivity to other Availability Zones in the same Region.

AWS Cloud Adoption Framework (AWS CAF)

A framework of guidelines and best practices from AWS to help organizations develop an efficient and effective plan to move successfully to the cloud. AWS CAF organizes guidance into six focus areas called perspectives: business, people, governance, platform, security, and operations. The business, people, and governance perspectives focus on business skills and processes; the platform, security, and operations perspectives focus on technical skills and processes. For example, the people perspective targets stakeholders who handle human resources (HR), staffing functions, and people management. For this perspective, AWS CAF provides guidance for people development, training, and communications to help ready the organization for successful cloud adoption. For more information, see the AWS CAF website and the AWS CAF whitepaper.

AWS Workload Qualification Framework (AWS WQF)

A tool that evaluates database migration workloads, recommends migration strategies, and provides work estimates. AWS WQF is included with AWS Schema Conversion Tool (AWS SCT). It analyzes database schemas and code objects, application code, dependencies, and performance characteristics, and provides assessment reports.

B

bad bot

A bot that is intended to disrupt or cause harm to individuals or organizations.

BCP

See business continuity planning.

behavior graph

A unified, interactive view of resource behavior and interactions over time. You can use a behavior graph with Amazon Detective to examine failed logon attempts, suspicious API calls, and similar actions. For more information, see Data in a behavior graph in the Detective documentation.

big-endian system

A system that stores the most significant byte first. See also endianness.

binary classification

A process that predicts a binary outcome (one of two possible classes). For example, your ML model might need to predict problems such as “Is this email spam or not spam?" or "Is this product a book or a car?"

bloom filter

A probabilistic, memory-efficient data structure that is used to test whether an element is a member of a set.

blue/green deployment

A deployment strategy where you create two separate but identical environments. You run the current application version in one environment (blue) and the new application version in the other environment (green). This strategy helps you quickly roll back with minimal impact.

bot

A software application that runs automated tasks over the internet and simulates human activity or interaction. Some bots are useful or beneficial, such as web crawlers that index information on the internet. Some other bots, known as *bad bots*, are intended to disrupt or cause harm to individuals or organizations.

botnet

Networks of bots that are infected by malware and are under the control of a single party, known as a *bot herder* or *bot operator*. Botnets are the best-known mechanism to scale bots and their impact.

branch

A contained area of a code repository. The first branch created in a repository is the *main branch*. You can create a new branch from an existing branch, and you can then develop features or fix bugs in the new branch. A branch you create to build a feature is commonly referred to as a *feature branch*. When the feature is ready for release, you merge the feature branch back into the main branch. For more information, see About branches (GitHub documentation).

break-glass access

In exceptional circumstances and through an approved process, a quick means for a user to gain access to an AWS account that they don't typically have permissions to access. For more information, see the Implement break-glass procedures indicator in the AWS Well-Architected guidance.

brownfield strategy

The existing infrastructure in your environment. When adopting a brownfield strategy for a system architecture, you design the architecture around the constraints of the current systems and infrastructure. If you are expanding the existing infrastructure, you might blend brownfield and greenfield strategies.

buffer cache

The memory area where the most frequently accessed data is stored.

business capability

What a business does to generate value (for example, sales, customer service, or marketing). Microservices architectures and development decisions can be driven by business capabilities. For more information, see the Organized around business capabilities section of the Running containerized microservices on AWS whitepaper.

business continuity planning (BCP)

A plan that addresses the potential impact of a disruptive event, such as a large-scale migration, on operations and enables a business to resume operations quickly.

C

CAF

See AWS Cloud Adoption Framework.

canary deployment

The slow and incremental release of a version to end users. When you are confident, you deploy the new version and replace the current version in its entirety.

CCoE

See Cloud Center of Excellence.

CDC

See change data capture.

change data capture (CDC)

The process of tracking changes to a data source, such as a database table, and recording metadata about the change. You can use CDC for various purposes, such as auditing or replicating changes in a target system to maintain synchronization.

chaos engineering

Intentionally introducing failures or disruptive events to test a system's resilience. You can use AWS Fault Injection Service (AWS FIS) to perform experiments that stress your AWS workloads and evaluate their response.

CI/CD

See continuous integration and continuous delivery.

classification

A categorization process that helps generate predictions. ML models for classification problems predict a discrete value. Discrete values are always distinct from one another. For example, a model might need to evaluate whether or not there is a car in an image.

client-side encryption

Encryption of data locally, before the target AWS service receives it.

Cloud Center of Excellence (CCoE)

A multi-disciplinary team that drives cloud adoption efforts across an organization, including developing cloud best practices, mobilizing resources, establishing migration timelines, and leading the organization through large-scale transformations. For more information, see the CCoE posts on the AWS Cloud Enterprise Strategy Blog.

cloud computing

The cloud technology that is typically used for remote data storage and IoT device management. Cloud computing is commonly connected to edge computing technology.

cloud operating model

In an IT organization, the operating model that is used to build, mature, and optimize one or more cloud environments. For more information, see Building your Cloud Operating Model.

cloud stages of adoption

The four phases that organizations typically go through when they migrate to the AWS Cloud:

- Project – Running a few cloud-related projects for proof of concept and learning purposes
- Foundation – Making foundational investments to scale your cloud adoption (e.g., creating a landing zone, defining a CCoE, establishing an operations model)
- Migration – Migrating individual applications
- Re-invention – Optimizing products and services, and innovating in the cloud

These stages were defined by Stephen Orban in the blog post The Journey Toward Cloud-First & the Stages of Adoption on the AWS Cloud Enterprise Strategy blog. For information about how they relate to the AWS migration strategy, see the migration readiness guide.

CMDB

See configuration management database.

code repository

A location where source code and other assets, such as documentation, samples, and scripts, are stored and updated through version control processes. Common cloud repositories include GitHub or Bitbucket Cloud. Each version of the code is called a *branch*. In a microservice structure, each repository is devoted to a single piece of functionality. A single CI/CD pipeline can use multiple repositories.

cold cache

A buffer cache that is empty, not well populated, or contains stale or irrelevant data. This affects performance because the database instance must read from the main memory or disk, which is slower than reading from the buffer cache.

cold data

Data that is rarely accessed and is typically historical. When querying this kind of data, slow queries are typically acceptable. Moving this data to lower-performing and less expensive storage tiers or classes can reduce costs.

computer vision (CV)

A field of AI that uses machine learning to analyze and extract information from visual formats such as digital images and videos. For example, Amazon SageMaker AI provides image processing algorithms for CV.

configuration drift

For a workload, a configuration change from the expected state. It might cause the workload to become noncompliant, and it's typically gradual and unintentional.

configuration management database (CMDB)

A repository that stores and manages information about a database and its IT environment, including both hardware and software components and their configurations. You typically use data from a CMDB in the portfolio discovery and analysis stage of migration.

conformance pack

A collection of AWS Config rules and remediation actions that you can assemble to customize your compliance and security checks. You can deploy a conformance pack as a single entity in an AWS account and Region, or across an organization, by using a YAML template. For more information, see Conformance packs in the AWS Config documentation.

continuous integration and continuous delivery (CI/CD)

The process of automating the source, build, test, staging, and production stages of the software release process. CI/CD is commonly described as a pipeline. CI/CD can help you automate processes, improve productivity, improve code quality, and deliver faster. For more information, see Benefits of continuous delivery. CD can also stand for *continuous deployment*. For more information, see Continuous Delivery vs. Continuous Deployment.

CV

See computer vision.

D

data at rest

Data that is stationary in your network, such as data that is in storage.

data classification

A process for identifying and categorizing the data in your network based on its criticality and sensitivity. It is a critical component of any cybersecurity risk management strategy because it helps you determine the appropriate protection and retention controls for the data. Data classification is a component of the security pillar in the AWS Well-Architected Framework. For more information, see Data classification.

data drift

A meaningful variation between the production data and the data that was used to train an ML model, or a meaningful change in the input data over time. Data drift can reduce the overall quality, accuracy, and fairness in ML model predictions.

data in transit

Data that is actively moving through your network, such as between network resources.

data mesh

An architectural framework that provides distributed, decentralized data ownership with centralized management and governance.

data minimization

The principle of collecting and processing only the data that is strictly necessary. Practicing data minimization in the AWS Cloud can reduce privacy risks, costs, and your analytics carbon footprint.

data perimeter

A set of preventive guardrails in your AWS environment that help make sure that only trusted identities are accessing trusted resources from expected networks. For more information, see Building a data perimeter on AWS.

data preprocessing

To transform raw data into a format that is easily parsed by your ML model. Preprocessing data can mean removing certain columns or rows and addressing missing, inconsistent, or duplicate values.

data provenance

The process of tracking the origin and history of data throughout its lifecycle, such as how the data was generated, transmitted, and stored.

data subject

An individual whose data is being collected and processed.

data warehouse

A data management system that supports business intelligence, such as analytics. Data warehouses commonly contain large amounts of historical data, and they are typically used for queries and analysis.

database definition language (DDL)

Statements or commands for creating or modifying the structure of tables and objects in a database.

database manipulation language (DML)

Statements or commands for modifying (inserting, updating, and deleting) information in a database.

DDL

See database definition language.

deep ensemble

To combine multiple deep learning models for prediction. You can use deep ensembles to obtain a more accurate prediction or for estimating uncertainty in predictions.

deep learning

An ML subfield that uses multiple layers of artificial neural networks to identify mapping between input data and target variables of interest.

defense-in-depth

An information security approach in which a series of security mechanisms and controls are thoughtfully layered throughout a computer network to protect the confidentiality, integrity, and availability of the network and the data within. When you adopt this strategy on AWS, you add multiple controls at different layers of the AWS Organizations structure to help secure resources. For example, a defense-in-depth approach might combine multi-factor authentication, network segmentation, and encryption.

delegated administrator

In AWS Organizations, a compatible service can register an AWS member account to administer the organization's accounts and manage permissions for that service. This account is called the *delegated administrator* for that service. For more information and a list of compatible services, see Services that work with AWS Organizations in the AWS Organizations documentation.

deployment

The process of making an application, new features, or code fixes available in the target environment. Deployment involves implementing changes in a code base and then building and running that code base in the application's environments.

development environment

See environment.

detective control

A security control that is designed to detect, log, and alert after an event has occurred. These controls are a second line of defense, alerting you to security events that bypassed the preventative controls in place. For more information, see Detective controls in *Implementing security controls on AWS*.

development value stream mapping (DVSM)

A process used to identify and prioritize constraints that adversely affect speed and quality in a software development lifecycle. DVSM extends the value stream mapping process originally designed for lean manufacturing practices. It focuses on the steps and teams required to create and move value through the software development process.

digital twin

A virtual representation of a real-world system, such as a building, factory, industrial equipment, or production line. Digital twins support predictive maintenance, remote monitoring, and production optimization.

dimension table

In a star schema, a smaller table that contains data attributes about quantitative data in a fact table. Dimension table attributes are typically text fields or discrete numbers that behave like text. These attributes are commonly used for query constraining, filtering, and result set labeling.

disaster

An event that prevents a workload or system from fulfilling its business objectives in its primary deployed location. These events can be natural disasters, technical failures, or the result of human actions, such as unintentional misconfiguration or a malware attack.

disaster recovery (DR)

The strategy and process you use to minimize downtime and data loss caused by a disaster. For more information, see Disaster Recovery of Workloads on AWS: Recovery in the Cloud in the AWS Well-Architected Framework.

DML

See database manipulation language.

domain-driven design

An approach to developing a complex software system by connecting its components to evolving domains, or core business goals, that each component serves. This concept was introduced by Eric Evans in his book, *Domain-Driven Design: Tackling Complexity in the Heart of Software* (Boston: Addison-Wesley Professional, 2003). For information about how you can use domain-driven design with the strangler fig pattern, see Modernizing legacy Microsoft ASP.NET (ASMX) web services incrementally by using containers and Amazon API Gateway.

DR

See disaster recovery.

drift detection

Tracking deviations from a baselined configuration. For example, you can use AWS CloudFormation to detect drift in system resources, or you can use AWS Control Tower to detect changes in your landing zone that might affect compliance with governance requirements.

DVSM

See development value stream mapping.

E

EDA

See exploratory data analysis.

EDI

See electronic data interchange.

edge computing

The technology that increases the computing power for smart devices at the edges of an IoT network. When compared with cloud computing, edge computing can reduce communication latency and improve response time.

electronic data interchange (EDI)

The automated exchange of business documents between organizations. For more information, see What is Electronic Data Interchange.

encryption

A computing process that transforms plaintext data, which is human-readable, into ciphertext.

encryption key

A cryptographic string of randomized bits that is generated by an encryption algorithm. Keys can vary in length, and each key is designed to be unpredictable and unique.

endianness

The order in which bytes are stored in computer memory. Big-endian systems store the most significant byte first. Little-endian systems store the least significant byte first.

endpoint

See service endpoint.

endpoint service

A service that you can host in a virtual private cloud (VPC) to share with other users. You can create an endpoint service with AWS PrivateLink and grant permissions to other AWS accounts or to AWS Identity and Access Management (IAM) principals. These accounts or principals can connect to your endpoint service privately by creating interface VPC endpoints. For more information, see Create an endpoint service in the Amazon Virtual Private Cloud (Amazon VPC) documentation.

enterprise resource planning (ERP)

A system that automates and manages key business processes (such as accounting, MES, and project management) for an enterprise.

envelope encryption

The process of encrypting an encryption key with another encryption key. For more information, see Envelope encryption in the AWS Key Management Service (AWS KMS) documentation.

environment

An instance of a running application. The following are common types of environments in cloud computing:

- development environment – An instance of a running application that is available only to the core team responsible for maintaining the application. Development environments are used to test changes before promoting them to upper environments. This type of environment is sometimes referred to as a *test environment*.
- lower environments – All development environments for an application, such as those used for initial builds and tests.
- production environment – An instance of a running application that end users can access. In a CI/CD pipeline, the production environment is the last deployment environment.
- upper environments – All environments that can be accessed by users other than the core development team. This can include a production environment, preproduction environments, and environments for user acceptance testing.

epic

In agile methodologies, functional categories that help organize and prioritize your work. Epics provide a high-level description of requirements and implementation tasks. For example, AWS CAF security epics include identity and access management, detective controls, infrastructure security, data protection, and incident response. For more information about epics in the AWS migration strategy, see the program implementation guide.

ERP

See enterprise resource planning.

exploratory data analysis (EDA)

The process of analyzing a dataset to understand its main characteristics. You collect or aggregate data and then perform initial investigations to find patterns, detect anomalies, and check assumptions. EDA is performed by calculating summary statistics and creating data visualizations.

F

fact table

The central table in a star schema. It stores quantitative data about business operations. Typically, a fact table contains two types of columns: those that contain measures and those that contain a foreign key to a dimension table.

fail fast

A philosophy that uses frequent and incremental testing to reduce the development lifecycle. It is a critical part of an agile approach.

fault isolation boundary

In the AWS Cloud, a boundary such as an Availability Zone, AWS Region, control plane, or data plane that limits the effect of a failure and helps improve the resilience of workloads. For more information, see AWS Fault Isolation Boundaries.

feature branch

See branch.

features

The input data that you use to make a prediction. For example, in a manufacturing context, features could be images that are periodically captured from the manufacturing line.

feature importance

How significant a feature is for a model's predictions. This is usually expressed as a numerical score that can be calculated through various techniques, such as Shapley Additive Explanations (SHAP) and integrated gradients. For more information, see Machine learning model interpretability with AWS.

feature transformation

To optimize data for the ML process, including enriching data with additional sources, scaling values, or extracting multiple sets of information from a single data field. This enables the ML model to benefit from the data. For example, if you break down the "2021-05-27 00:15:37" date into "2021", "May", "Thu", and "15", you can help the learning algorithm learn nuanced patterns associated with different data components.

few-shot prompting

Providing an LLM with a small number of examples that demonstrate the task and desired output before asking it to perform a similar task. This technique is an application of in-context learning, where models learn from examples (*shots*) that are embedded in prompts. Few-shot prompting can be effective for tasks that require specific formatting, reasoning, or domain knowledge. See also zero-shot prompting.

FGAC

See fine-grained access control.

fine-grained access control (FGAC)

The use of multiple conditions to allow or deny an access request.

flash-cut migration

A database migration method that uses continuous data replication through change data capture to migrate data in the shortest time possible, instead of using a phased approach. The objective is to keep downtime to a minimum.

FM

See foundation model.

foundation model (FM)

A large deep-learning neural network that has been training on massive datasets of generalized and unlabeled data. FMs are capable of performing a wide variety of general tasks, such as understanding language, generating text and images, and conversing in natural language. For more information, see What are Foundation Models.

G

generative AI

A subset of AI models that have been trained on large amounts of data and that can use a simple text prompt to create new content and artifacts, such as images, videos, text, and audio. For more information, see What is Generative AI.

geo blocking

See geographic restrictions.

geographic restrictions (geo blocking)

In Amazon CloudFront, an option to prevent users in specific countries from accessing content distributions. You can use an allow list or block list to specify approved and banned countries. For more information, see Restricting the geographic distribution of your content in the CloudFront documentation.

Gitflow workflow

An approach in which lower and upper environments use different branches in a source code repository. The Gitflow workflow is considered legacy, and the trunk-based workflow is the modern, preferred approach.

golden image

A snapshot of a system or software that is used as a template to deploy new instances of that system or software. For example, in manufacturing, a golden image can be used to provision software on multiple devices and helps improve speed, scalability, and productivity in device manufacturing operations.

greenfield strategy

The absence of existing infrastructure in a new environment. When adopting a greenfield strategy for a system architecture, you can select all new technologies without the restriction

of compatibility with existing infrastructure, also known as brownfield. If you are expanding the existing infrastructure, you might blend brownfield and greenfield strategies.

guardrail

A high-level rule that helps govern resources, policies, and compliance across organizational units (OUs). *Preventive guardrails* enforce policies to ensure alignment to compliance standards. They are implemented by using service control policies and IAM permissions boundaries. *Detective guardrails* detect policy violations and compliance issues, and generate alerts for remediation. They are implemented by using AWS Config, AWS Security Hub, Amazon GuardDuty, AWS Trusted Advisor, Amazon Inspector, and custom AWS Lambda checks.

H

HA

See high availability.

heterogeneous database migration

Migrating your source database to a target database that uses a different database engine (for example, Oracle to Amazon Aurora). Heterogeneous migration is typically part of a re-architecting effort, and converting the schema can be a complex task. AWS provides AWS SCT that helps with schema conversions.

high availability (HA)

The ability of a workload to operate continuously, without intervention, in the event of challenges or disasters. HA systems are designed to automatically fail over, consistently deliver high-quality performance, and handle different loads and failures with minimal performance impact.

historian modernization

An approach used to modernize and upgrade operational technology (OT) systems to better serve the needs of the manufacturing industry. A *historian* is a type of database that is used to collect and store data from various sources in a factory.

holdout data

A portion of historical, labeled data that is withheld from a dataset that is used to train a machine learning model. You can use holdout data to evaluate the model performance by comparing the model predictions against the holdout data.

homogeneous database migration

Migrating your source database to a target database that shares the same database engine (for example, Microsoft SQL Server to Amazon RDS for SQL Server). Homogeneous migration is typically part of a rehosting or replatforming effort. You can use native database utilities to migrate the schema.

hot data

Data that is frequently accessed, such as real-time data or recent translational data. This data typically requires a high-performance storage tier or class to provide fast query responses.

hotfix

An urgent fix for a critical issue in a production environment. Due to its urgency, a hotfix is usually made outside of the typical DevOps release workflow.

hypercare period

Immediately following cutover, the period of time when a migration team manages and monitors the migrated applications in the cloud in order to address any issues. Typically, this period is 1–4 days in length. At the end of the hypercare period, the migration team typically transfers responsibility for the applications to the cloud operations team.

I

IaC

See infrastructure as code.

identity-based policy

A policy attached to one or more IAM principals that defines their permissions within the AWS Cloud environment.

idle application

An application that has an average CPU and memory usage between 5 and 20 percent over a period of 90 days. In a migration project, it is common to retire these applications or retain them on premises.

IIoT

See industrial Internet of Things.

immutable infrastructure

A model that deploys new infrastructure for production workloads instead of updating, patching, or modifying the existing infrastructure. Immutable infrastructures are inherently more consistent, reliable, and predictable than mutable infrastructure. For more information, see the Deploy using immutable infrastructure best practice in the AWS Well-Architected Framework.

inbound (ingress) VPC

In an AWS multi-account architecture, a VPC that accepts, inspects, and routes network connections from outside an application. The AWS Security Reference Architecture recommends setting up your Network account with inbound, outbound, and inspection VPCs to protect the two-way interface between your application and the broader internet.

incremental migration

A cutover strategy in which you migrate your application in small parts instead of performing a single, full cutover. For example, you might move only a few microservices or users to the new system initially. After you verify that everything is working properly, you can incrementally move additional microservices or users until you can decommission your legacy system. This strategy reduces the risks associated with large migrations.

Industry 4.0

A term that was introduced by Klaus Schwab in 2016 to refer to the modernization of manufacturing processes through advances in connectivity, real-time data, automation, analytics, and AI/ML.

infrastructure

All of the resources and assets contained within an application's environment.

infrastructure as code (IaC)

The process of provisioning and managing an application's infrastructure through a set of configuration files. IaC is designed to help you centralize infrastructure management, standardize resources, and scale quickly so that new environments are repeatable, reliable, and consistent.

industrial Internet of Things (IIoT)

The use of internet-connected sensors and devices in the industrial sectors, such as manufacturing, energy, automotive, healthcare, life sciences, and agriculture. For more information, see Building an industrial Internet of Things (IIoT) digital transformation strategy.

inspection VPC

In an AWS multi-account architecture, a centralized VPC that manages inspections of network traffic between VPCs (in the same or different AWS Regions), the internet, and on-premises networks. The AWS Security Reference Architecture recommends setting up your Network account with inbound, outbound, and inspection VPCs to protect the two-way interface between your application and the broader internet.

Internet of Things (IoT)

The network of connected physical objects with embedded sensors or processors that communicate with other devices and systems through the internet or over a local communication network. For more information, see What is IoT?

interpretability

A characteristic of a machine learning model that describes the degree to which a human can understand how the model's predictions depend on its inputs. For more information, see Machine learning model interpretability with AWS.

IoT

See Internet of Things.

IT information library (ITIL)

A set of best practices for delivering IT services and aligning these services with business requirements. ITIL provides the foundation for ITSM.

IT service management (ITSM)

Activities associated with designing, implementing, managing, and supporting IT services for an organization. For information about integrating cloud operations with ITSM tools, see the operations integration guide.

ITIL

See IT information library.

ITSM

See IT service management.

L

label-based access control (LBAC)

An implementation of mandatory access control (MAC) where the users and the data itself are each explicitly assigned a security label value. The intersection between the user security label and data security label determines which rows and columns can be seen by the user.

landing zone

A landing zone is a well-architected, multi-account AWS environment that is scalable and secure. This is a starting point from which your organizations can quickly launch and deploy workloads and applications with confidence in their security and infrastructure environment. For more information about landing zones, see Setting up a secure and scalable multi-account AWS environment.

large language model (LLM)

A deep learning AI model that is pretrained on a vast amount of data. An LLM can perform multiple tasks, such as answering questions, summarizing documents, translating text into other languages, and completing sentences. For more information, see What are LLMs.

large migration

A migration of 300 or more servers.

LBAC

See label-based access control.

least privilege

The security best practice of granting the minimum permissions required to perform a task. For more information, see Apply least-privilege permissions in the IAM documentation.

lift and shift

See 7 Rs.

little-endian system

A system that stores the least significant byte first. See also endianness.

LLM

See large language model.

lower environments

See environment.

M

machine learning (ML)

A type of artificial intelligence that uses algorithms and techniques for pattern recognition and learning. ML analyzes and learns from recorded data, such as Internet of Things (IoT) data, to generate a statistical model based on patterns. For more information, see Machine Learning.

main branch

See branch.

malware

Software that is designed to compromise computer security or privacy. Malware might disrupt computer systems, leak sensitive information, or gain unauthorized access. Examples of malware include viruses, worms, ransomware, Trojan horses, spyware, and keyloggers.

managed services

AWS services for which AWS operates the infrastructure layer, the operating system, and platforms, and you access the endpoints to store and retrieve data. Amazon Simple Storage Service (Amazon S3) and Amazon DynamoDB are examples of managed services. These are also known as *abstracted services*.

manufacturing execution system (MES)

A software system for tracking, monitoring, documenting, and controlling production processes that convert raw materials to finished products on the shop floor.

MAP

See Migration Acceleration Program.

mechanism

A complete process in which you create a tool, drive adoption of the tool, and then inspect the results in order to make adjustments. A mechanism is a cycle that reinforces and improves itself as it operates. For more information, see Building mechanisms in the AWS Well-Architected Framework.

member account

All AWS accounts other than the management account that are part of an organization in AWS Organizations. An account can be a member of only one organization at a time.

MES

See manufacturing execution system.

Message Queuing Telemetry Transport (MQTT)

A lightweight, machine-to-machine (M2M) communication protocol, based on the publish/subscribe pattern, for resource-constrained IoT devices.

microservice

A small, independent service that communicates over well-defined APIs and is typically owned by small, self-contained teams. For example, an insurance system might include microservices that map to business capabilities, such as sales or marketing, or subdomains, such as purchasing, claims, or analytics. The benefits of microservices include agility, flexible scaling, easy deployment, reusable code, and resilience. For more information, see Integrating microservices by using AWS serverless services.

microservices architecture

An approach to building an application with independent components that run each application process as a microservice. These microservices communicate through a well-defined interface by using lightweight APIs. Each microservice in this architecture can be updated, deployed,

and scaled to meet demand for specific functions of an application. For more information, see Implementing microservices on AWS.

Migration Acceleration Program (MAP)

An AWS program that provides consulting support, training, and services to help organizations build a strong operational foundation for moving to the cloud, and to help offset the initial cost of migrations. MAP includes a migration methodology for executing legacy migrations in a methodical way and a set of tools to automate and accelerate common migration scenarios.

migration at scale

The process of moving the majority of the application portfolio to the cloud in waves, with more applications moved at a faster rate in each wave. This phase uses the best practices and lessons learned from the earlier phases to implement a *migration factory* of teams, tools, and processes to streamline the migration of workloads through automation and agile delivery. This is the third phase of the AWS migration strategy.

migration factory

Cross-functional teams that streamline the migration of workloads through automated, agile approaches. Migration factory teams typically include operations, business analysts and owners, migration engineers, developers, and DevOps professionals working in sprints. Between 20 and 50 percent of an enterprise application portfolio consists of repeated patterns that can be optimized by a factory approach. For more information, see the discussion of migration factories and the Cloud Migration Factory guide in this content set.

migration metadata

The information about the application and server that is needed to complete the migration. Each migration pattern requires a different set of migration metadata. Examples of migration metadata include the target subnet, security group, and AWS account.

migration pattern

A repeatable migration task that details the migration strategy, the migration destination, and the migration application or service used. Example: Rehost migration to Amazon EC2 with AWS Application Migration Service.

Migration Portfolio Assessment (MPA)

An online tool that provides information for validating the business case for migrating to the AWS Cloud. MPA provides detailed portfolio assessment (server right-sizing, pricing, TCO

comparisons, migration cost analysis) as well as migration planning (application data analysis and data collection, application grouping, migration prioritization, and wave planning). The MPA tool (requires login) is available free of charge to all AWS consultants and APN Partner consultants.

Migration Readiness Assessment (MRA)

The process of gaining insights about an organization's cloud readiness status, identifying strengths and weaknesses, and building an action plan to close identified gaps, using the AWS CAF. For more information, see the migration readiness guide. MRA is the first phase of the AWS migration strategy.

migration strategy

The approach used to migrate a workload to the AWS Cloud. For more information, see the 7 Rs entry in this glossary and see Mobilize your organization to accelerate large-scale migrations.

ML

See machine learning.

modernization

Transforming an outdated (legacy or monolithic) application and its infrastructure into an agile, elastic, and highly available system in the cloud to reduce costs, gain efficiencies, and take advantage of innovations. For more information, see Strategy for modernizing applications in the AWS Cloud.

modernization readiness assessment

An evaluation that helps determine the modernization readiness of an organization's applications; identifies benefits, risks, and dependencies; and determines how well the organization can support the future state of those applications. The outcome of the assessment is a blueprint of the target architecture, a roadmap that details development phases and milestones for the modernization process, and an action plan for addressing identified gaps. For more information, see Evaluating modernization readiness for applications in the AWS Cloud.

monolithic applications (monoliths)

Applications that run as a single service with tightly coupled processes. Monolithic applications have several drawbacks. If one application feature experiences a spike in demand, the entire architecture must be scaled. Adding or improving a monolithic application's features also becomes more complex when the code base grows. To address these issues, you can

use a microservices architecture. For more information, see Decomposing monoliths into microservices.

MPA

See Migration Portfolio Assessment.

MQTT

See Message Queuing Telemetry Transport.

multiclass classification

A process that helps generate predictions for multiple classes (predicting one of more than two outcomes). For example, an ML model might ask "Is this product a book, car, or phone?" or "Which product category is most interesting to this customer?"

mutable infrastructure

A model that updates and modifies the existing infrastructure for production workloads. For improved consistency, reliability, and predictability, the AWS Well-Architected Framework recommends the use of immutable infrastructure as a best practice.

O

OAC

See origin access control.

OAI

See origin access identity.

OCM

See organizational change management.

offline migration

A migration method in which the source workload is taken down during the migration process. This method involves extended downtime and is typically used for small, non-critical workloads.

OI

See operations integration.

OLA

See operational-level agreement.

online migration

A migration method in which the source workload is copied to the target system without being taken offline. Applications that are connected to the workload can continue to function during the migration. This method involves zero to minimal downtime and is typically used for critical production workloads.

OPC-UA

See Open Process Communications - Unified Architecture.

Open Process Communications - Unified Architecture (OPC-UA)

A machine-to-machine (M2M) communication protocol for industrial automation. OPC-UA provides an interoperability standard with data encryption, authentication, and authorization schemes.

operational-level agreement (OLA)

An agreement that clarifies what functional IT groups promise to deliver to each other, to support a service-level agreement (SLA).

operational readiness review (ORR)

A checklist of questions and associated best practices that help you understand, evaluate, prevent, or reduce the scope of incidents and possible failures. For more information, see Operational Readiness Reviews (ORR) in the AWS Well-Architected Framework.

operational technology (OT)

Hardware and software systems that work with the physical environment to control industrial operations, equipment, and infrastructure. In manufacturing, the integration of OT and information technology (IT) systems is a key focus for Industry 4.0 transformations.

operations integration (OI)

The process of modernizing operations in the cloud, which involves readiness planning, automation, and integration. For more information, see the operations integration guide.

organization trail

A trail that's created by AWS CloudTrail that logs all events for all AWS accounts in an organization in AWS Organizations. This trail is created in each AWS account that's part of the

organization and tracks the activity in each account. For more information, see Creating a trail for an organization in the CloudTrail documentation.

organizational change management (OCM)

A framework for managing major, disruptive business transformations from a people, culture, and leadership perspective. OCM helps organizations prepare for, and transition to, new systems and strategies by accelerating change adoption, addressing transitional issues, and driving cultural and organizational changes. In the AWS migration strategy, this framework is called *people acceleration*, because of the speed of change required in cloud adoption projects. For more information, see the OCM guide.

origin access control (OAC)

In CloudFront, an enhanced option for restricting access to secure your Amazon Simple Storage Service (Amazon S3) content. OAC supports all S3 buckets in all AWS Regions, server-side encryption with AWS KMS (SSE-KMS), and dynamic PUT and DELETE requests to the S3 bucket.

origin access identity (OAI)

In CloudFront, an option for restricting access to secure your Amazon S3 content. When you use OAI, CloudFront creates a principal that Amazon S3 can authenticate with. Authenticated principals can access content in an S3 bucket only through a specific CloudFront distribution. See also OAC, which provides more granular and enhanced access control.

ORR

See operational readiness review.

OT

See operational technology.

outbound (egress) VPC

In an AWS multi-account architecture, a VPC that handles network connections that are initiated from within an application. The AWS Security Reference Architecture recommends setting up your Network account with inbound, outbound, and inspection VPCs to protect the two-way interface between your application and the broader internet.

P

permissions boundary

An IAM management policy that is attached to IAM principals to set the maximum permissions that the user or role can have. For more information, see Permissions boundaries in the IAM documentation.

personally identifiable information (PII)

Information that, when viewed directly or paired with other related data, can be used to reasonably infer the identity of an individual. Examples of PII include names, addresses, and contact information.

PII

See personally identifiable information.

playbook

A set of predefined steps that capture the work associated with migrations, such as delivering core operations functions in the cloud. A playbook can take the form of scripts, automated runbooks, or a summary of processes or steps required to operate your modernized environment.

PLC

See programmable logic controller.

PLM

See product lifecycle management.

policy

An object that can define permissions (see identity-based policy), specify access conditions (see resource-based policy), or define the maximum permissions for all accounts in an organization in AWS Organizations (see service control policy).

polyglot persistence

Independently choosing a microservice's data storage technology based on data access patterns and other requirements. If your microservices have the same data storage technology, they can encounter implementation challenges or experience poor performance. Microservices are more easily implemented and achieve better performance and scalability if they use the data store

best adapted to their requirements. For more information, see Enabling data persistence in microservices.

portfolio assessment

A process of discovering, analyzing, and prioritizing the application portfolio in order to plan the migration. For more information, see Evaluating migration readiness.

predicate

A query condition that returns `true` or `false`, commonly located in a `WHERE` clause.

predicate pushdown

A database query optimization technique that filters the data in the query before transfer. This reduces the amount of data that must be retrieved and processed from the relational database, and it improves query performance.

preventative control

A security control that is designed to prevent an event from occurring. These controls are a first line of defense to help prevent unauthorized access or unwanted changes to your network. For more information, see Preventative controls in *Implementing security controls on AWS*.

principal

An entity in AWS that can perform actions and access resources. This entity is typically a root user for an AWS account, an IAM role, or a user. For more information, see *Principal* in Roles terms and concepts in the IAM documentation.

privacy by design

A system engineering approach that takes privacy into account through the whole development process.

private hosted zones

A container that holds information about how you want Amazon Route 53 to respond to DNS queries for a domain and its subdomains within one or more VPCs. For more information, see Working with private hosted zones in the Route 53 documentation.

proactive control

A security control designed to prevent the deployment of noncompliant resources. These controls scan resources before they are provisioned. If the resource is not compliant with the control, then it isn't provisioned. For more information, see the Controls reference guide in the

AWS Control Tower documentation and see Proactive controls in *Implementing security controls on AWS*.

product lifecycle management (PLM)

The management of data and processes for a product throughout its entire lifecycle, from design, development, and launch, through growth and maturity, to decline and removal.

production environment

See environment.

programmable logic controller (PLC)

In manufacturing, a highly reliable, adaptable computer that monitors machines and automates manufacturing processes.

prompt chaining

Using the output of one LLM prompt as the input for the next prompt to generate better responses. This technique is used to break down a complex task into subtasks, or to iteratively refine or expand a preliminary response. It helps improve the accuracy and relevance of a model's responses and allows for more granular, personalized results.

pseudonymization

The process of replacing personal identifiers in a dataset with placeholder values. Pseudonymization can help protect personal privacy. Pseudonymized data is still considered to be personal data.

publish/subscribe (pub/sub)

A pattern that enables asynchronous communications among microservices to improve scalability and responsiveness. For example, in a microservices-based MES, a microservice can publish event messages to a channel that other microservices can subscribe to. The system can add new microservices without changing the publishing service.

Q

query plan

A series of steps, like instructions, that are used to access the data in a SQL relational database system.

query plan regression

When a database service optimizer chooses a less optimal plan than it did before a given change to the database environment. This can be caused by changes to statistics, constraints, environment settings, query parameter bindings, and updates to the database engine.

R

RACI matrix

See responsible, accountable, consulted, informed (RACI).

RAG

See Retrieval Augmented Generation.

ransomware

A malicious software that is designed to block access to a computer system or data until a payment is made.

RASCI matrix

See responsible, accountable, consulted, informed (RACI).

RCAC

See row and column access control.

read replica

A copy of a database that's used for read-only purposes. You can route queries to the read replica to reduce the load on your primary database.

re-architect

See 7 Rs.

recovery point objective (RPO)

The maximum acceptable amount of time since the last data recovery point. This determines what is considered an acceptable loss of data between the last recovery point and the interruption of service.

recovery time objective (RTO)

The maximum acceptable delay between the interruption of service and restoration of service.

refactor

See 7 Rs.

Region

A collection of AWS resources in a geographic area. Each AWS Region is isolated and independent of the others to provide fault tolerance, stability, and resilience. For more information, see Specify which AWS Regions your account can use.

regression

An ML technique that predicts a numeric value. For example, to solve the problem of "What price will this house sell for?" an ML model could use a linear regression model to predict a house's sale price based on known facts about the house (for example, the square footage).

rehost

See 7 Rs.

release

In a deployment process, the act of promoting changes to a production environment.

relocate

See 7 Rs.

replatform

See 7 Rs.

repurchase

See 7 Rs.

resiliency

An application's ability to resist or recover from disruptions. High availability and disaster recovery are common considerations when planning for resiliency in the AWS Cloud. For more information, see AWS Cloud Resilience.

resource-based policy

A policy attached to a resource, such as an Amazon S3 bucket, an endpoint, or an encryption key. This type of policy specifies which principals are allowed access, supported actions, and any other conditions that must be met.

responsible, accountable, consulted, informed (RACI) matrix

A matrix that defines the roles and responsibilities for all parties involved in migration activities and cloud operations. The matrix name is derived from the responsibility types defined in the matrix: responsible (R), accountable (A), consulted (C), and informed (I). The support (S) type is optional. If you include support, the matrix is called a *RASCI matrix*, and if you exclude it, it's called a *RACI matrix*.

responsive control

A security control that is designed to drive remediation of adverse events or deviations from your security baseline. For more information, see Responsive controls in *Implementing security controls on AWS*.

retain

See 7 Rs.

retire

See 7 Rs.

Retrieval Augmented Generation (RAG)

A generative AI technology in which an LLM references an authoritative data source that is outside of its training data sources before generating a response. For example, a RAG model might perform a semantic search of an organization's knowledge base or custom data. For more information, see What is RAG.

rotation

The process of periodically updating a secret to make it more difficult for an attacker to access the credentials.

row and column access control (RCAC)

The use of basic, flexible SQL expressions that have defined access rules. RCAC consists of row permissions and column masks.

RPO

See recovery point objective.

RTO

See recovery time objective.

runbook

A set of manual or automated procedures required to perform a specific task. These are typically built to streamline repetitive operations or procedures with high error rates.

S

SAML 2.0

An open standard that many identity providers (IdPs) use. This feature enables federated single sign-on (SSO), so users can log into the AWS Management Console or call the AWS API operations without you having to create user in IAM for everyone in your organization. For more information about SAML 2.0-based federation, see About SAML 2.0-based federation in the IAM documentation.

SCADA

See supervisory control and data acquisition.

SCP

See service control policy.

secret

In AWS Secrets Manager, confidential or restricted information, such as a password or user credentials, that you store in encrypted form. It consists of the secret value and its metadata. The secret value can be binary, a single string, or multiple strings. For more information, see What's in a Secrets Manager secret? in the Secrets Manager documentation.

security by design

A system engineering approach that takes security into account through the whole development process.

security control

A technical or administrative guardrail that prevents, detects, or reduces the ability of a threat actor to exploit a security vulnerability. There are four primary types of security controls: preventative, detective, responsive, and proactive.

security hardening

The process of reducing the attack surface to make it more resistant to attacks. This can include actions such as removing resources that are no longer needed, implementing the security best practice of granting least privilege, or deactivating unnecessary features in configuration files.

security information and event management (SIEM) system

Tools and services that combine security information management (SIM) and security event management (SEM) systems. A SIEM system collects, monitors, and analyzes data from servers, networks, devices, and other sources to detect threats and security breaches, and to generate alerts.

security response automation

A predefined and programmed action that is designed to automatically respond to or remediate a security event. These automations serve as detective or responsive security controls that help you implement AWS security best practices. Examples of automated response actions include modifying a VPC security group, patching an Amazon EC2 instance, or rotating credentials.

server-side encryption

Encryption of data at its destination, by the AWS service that receives it.

service control policy (SCP)

A policy that provides centralized control over permissions for all accounts in an organization in AWS Organizations. SCPs define guardrails or set limits on actions that an administrator can delegate to users or roles. You can use SCPs as allow lists or deny lists, to specify which services or actions are permitted or prohibited. For more information, see Service control policies in the AWS Organizations documentation.

service endpoint

The URL of the entry point for an AWS service. You can use the endpoint to connect programmatically to the target service. For more information, see AWS service endpoints in *AWS General Reference*.

service-level agreement (SLA)

An agreement that clarifies what an IT team promises to deliver to their customers, such as service uptime and performance.

service-level indicator (SLI)

A measurement of a performance aspect of a service, such as its error rate, availability, or throughput.

service-level objective (SLO)

A target metric that represents the health of a service, as measured by a service-level indicator.

shared responsibility model

A model describing the responsibility you share with AWS for cloud security and compliance. AWS is responsible for security *of* the cloud, whereas you are responsible for security *in* the cloud. For more information, see Shared responsibility model.

SIEM

See security information and event management system.

single point of failure (SPOF)

A failure in a single, critical component of an application that can disrupt the system.

SLA

See service-level agreement.

SLI

See service-level indicator.

SLO

See service-level objective.

split-and-seed model

A pattern for scaling and accelerating modernization projects. As new features and product releases are defined, the core team splits up to create new product teams. This helps scale your organization's capabilities and services, improves developer productivity, and supports rapid

innovation. For more information, see Phased approach to modernizing applications in the AWS Cloud.

SPOF

See single point of failure.

star schema

A database organizational structure that uses one large fact table to store transactional or measured data and uses one or more smaller dimensional tables to store data attributes. This structure is designed for use in a data warehouse or for business intelligence purposes.

strangler fig pattern

An approach to modernizing monolithic systems by incrementally rewriting and replacing system functionality until the legacy system can be decommissioned. This pattern uses the analogy of a fig vine that grows into an established tree and eventually overcomes and replaces its host. The pattern was introduced by Martin Fowler as a way to manage risk when rewriting monolithic systems. For an example of how to apply this pattern, see Modernizing legacy Microsoft ASP.NET (ASMX) web services incrementally by using containers and Amazon API Gateway.

subnet

A range of IP addresses in your VPC. A subnet must reside in a single Availability Zone.

supervisory control and data acquisition (SCADA)

In manufacturing, a system that uses hardware and software to monitor physical assets and production operations.

symmetric encryption

An encryption algorithm that uses the same key to encrypt and decrypt the data.

synthetic testing

Testing a system in a way that simulates user interactions to detect potential issues or to monitor performance. You can use Amazon CloudWatch Synthetics to create these tests.

system prompt

A technique for providing context, instructions, or guidelines to an LLM to direct its behavior. System prompts help set context and establish rules for interactions with users.

T

tags

Key-value pairs that act as metadata for organizing your AWS resources. Tags can help you manage, identify, organize, search for, and filter resources. For more information, see Tagging your AWS resources.

target variable

The value that you are trying to predict in supervised ML. This is also referred to as an *outcome variable*. For example, in a manufacturing setting the target variable could be a product defect.

task list

A tool that is used to track progress through a runbook. A task list contains an overview of the runbook and a list of general tasks to be completed. For each general task, it includes the estimated amount of time required, the owner, and the progress.

test environment

See environment.

training

To provide data for your ML model to learn from. The training data must contain the correct answer. The learning algorithm finds patterns in the training data that map the input data attributes to the target (the answer that you want to predict). It outputs an ML model that captures these patterns. You can then use the ML model to make predictions on new data for which you don't know the target.

transit gateway

A network transit hub that you can use to interconnect your VPCs and on-premises networks. For more information, see What is a transit gateway in the AWS Transit Gateway documentation.

trunk-based workflow

An approach in which developers build and test features locally in a feature branch and then merge those changes into the main branch. The main branch is then built to the development, preproduction, and production environments, sequentially.

trusted access

Granting permissions to a service that you specify to perform tasks in your organization in AWS Organizations and in its accounts on your behalf. The trusted service creates a service-linked role in each account, when that role is needed, to perform management tasks for you. For more information, see Using AWS Organizations with other AWS services in the AWS Organizations documentation.

tuning

To change aspects of your training process to improve the ML model's accuracy. For example, you can train the ML model by generating a labeling set, adding labels, and then repeating these steps several times under different settings to optimize the model.

two-pizza team

A small DevOps team that you can feed with two pizzas. A two-pizza team size ensures the best possible opportunity for collaboration in software development.

U

uncertainty

A concept that refers to imprecise, incomplete, or unknown information that can undermine the reliability of predictive ML models. There are two types of uncertainty: *Epistemic uncertainty* is caused by limited, incomplete data, whereas *aleatoric uncertainty* is caused by the noise and randomness inherent in the data. For more information, see the Quantifying uncertainty in deep learning systems guide.

undifferentiated tasks

Also known as *heavy lifting*, work that is necessary to create and operate an application but that doesn't provide direct value to the end user or provide competitive advantage. Examples of undifferentiated tasks include procurement, maintenance, and capacity planning.

upper environments

See environment.

V

vacuuming

A database maintenance operation that involves cleaning up after incremental updates to reclaim storage and improve performance.

version control

Processes and tools that track changes, such as changes to source code in a repository.

VPC peering

A connection between two VPCs that allows you to route traffic by using private IP addresses. For more information, see What is VPC peering in the Amazon VPC documentation.

vulnerability

A software or hardware flaw that compromises the security of the system.

W

warm cache

A buffer cache that contains current, relevant data that is frequently accessed. The database instance can read from the buffer cache, which is faster than reading from the main memory or disk.

warm data

Data that is infrequently accessed. When querying this kind of data, moderately slow queries are typically acceptable.

window function

A SQL function that performs a calculation on a group of rows that relate in some way to the current record. Window functions are useful for processing tasks, such as calculating a moving average or accessing the value of rows based on the relative position of the current row.

workload

A collection of resources and code that delivers business value, such as a customer-facing application or backend process.

workstream

Functional groups in a migration project that are responsible for a specific set of tasks. Each workstream is independent but supports the other workstreams in the project. For example, the portfolio workstream is responsible for prioritizing applications, wave planning, and collecting migration metadata. The portfolio workstream delivers these assets to the migration workstream, which then migrates the servers and applications.

WORM

See write once, read many.

WQF

See AWS Workload Qualification Framework.

write once, read many (WORM)

A storage model that writes data a single time and prevents the data from being deleted or modified. Authorized users can read the data as many times as needed, but they cannot change it. This data storage infrastructure is considered immutable.

Z

zero-day exploit

An attack, typically malware, that takes advantage of a zero-day vulnerability.

zero-day vulnerability

An unmitigated flaw or vulnerability in a production system. Threat actors can use this type of vulnerability to attack the system. Developers frequently become aware of the vulnerability as a result of the attack.

zero-shot prompting

Providing an LLM with instructions for performing a task but no examples (*shots*) that can help guide it. The LLM must use its pre-trained knowledge to handle the task. The effectiveness of zero-shot prompting depends on the complexity of the task and the quality of the prompt. See also few-shot prompting.

zombie application

An application that has an average CPU and memory usage below 5 percent. In a migration project, it is common to retire these applications.