

Exam

AIP-C01

AWS Certified Generative AI Developer - Professional

Free demo	26 sample questions
Full exam	127 questions with answers and explanations
Contents	Questions with answers and explanations
Generated	21 October 2026

Exam domains

Domain	Exam weight	Questions
D1 Foundation Model (FM) Integration, Data Management, and Compliance	31%	10
D2 Implementation and Integration	26%	7
D3 AI Safety, Security, and Governance	18%	3
D4 Operational Efficiency and Optimization	12%	4
D5 Testing, Validation, and Troubleshooting	13%	2

This is a free demo from Dumpsgate: 26 of the exam's 127 questions. [Get the full AIP-C01 exam](#) with every question, answer and explanation.

Question 1

Single choice · Domain: D1 Foundation Model (FM) Integration, Data Management, and Compliance › Implement Retrieval Augmented Generation (RAG) solutions

A GenAI developer is building a RAG system that uses Amazon Bedrock Knowledge Bases. The system needs to process 50 textbooks that are stored in an Amazon S3 bucket. The textbooks are all an average of 500 pages long. The system needs to generate a knowledge base to answer domainspecific customer questions.

During initial testing on a subset of documents, the GenAI developer notices that query responses contain irrelevant information and sometimes miss critical context from the source materials. The GenAI developer must ensure that the solution provides accurate responses with low latency and no hallucinations.

Which solution will meet these requirements?

- A.** Configure fixed-size chunking with a 256-token chunk size. Implement metadata filtering based on document sections. Use an Amazon Titan Embeddings model to create vector representations that are optimized for semantic search.
- B.** Use Amazon ElastiCache to implement semantic caching for common queries. Use Anthropic Claude Sonnet to reformulate user queries. Configure real-time model feedback loops to continuously improve response quality based on user interactions.
- C.** Apply hierarchical chunking with both 200-token chunks and 1,000-token chunks. Implement hybrid search that combines vector and keyword search. Adjust the relevance score threshold to filter out low-confidence retrieval results.
- D.** Use semantic chunking to automatically segment documents based on topic boundaries. Configure the knowledge base to use a single embedding model for all content types. Implement query expansion to reformulate user questions before retrieval.

Answer: C. Apply hierarchical chunking with both 200-token chunks and 1,000-token chunks. Implement hybrid search that combines vector and keyword search. Adjust the relevance score threshold to filter out low-confidence retrieval results.

Explanation

Hierarchical chunking retrieves a focused 200-token child chunk while retaining a larger 1,000-token parent chunk for the surrounding context, which can help answer detailed questions about long textbooks without losing important source material. Hybrid search combines vector and keyword retrieval, helping find both semantically related passages and passages containing important terms. A relevance score threshold can exclude low-confidence results, reducing irrelevant context that could lead to inaccurate or unsupported answers. Together, these retrieval choices best address the observed relevance and context problems in the scenario, although no retrieval configuration can guarantee zero hallucinations. [8, 10, 11]

https://docs.aws.amazon.com/bedrock/latest/APIReference/API_agent_ChunkingConfiguration.html

<https://docs.aws.amazon.com/bedrock/latest/userguide/kb-test-config.html>

https://docs.aws.amazon.com/bedrock/latest/userguide/kb-chunking.html?trk=article-ssr-frontend-pulse_little-text-block

✗ **A** Configure fixed-size chunking with a 256-token chunk size. Implement metadata filtering based on document sections. Use an Amazon Titan Embeddings model to create vector representations that are optimized for semantic search.

A fixed 256-token chunk can separate a passage from surrounding context that is needed to answer a textbook question, and vector-only search can miss matches that depend on specific terms. Section

metadata filtering can narrow the search, but it does not provide the combination of multi-scale context and keyword retrieval offered by option C. [2, 5, 11]

<https://docs.aws.amazon.com/bedrock/latest/userguide/kb-data-source-customize-ingestion.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/titan-models.html>

https://docs.aws.amazon.com/bedrock/latest/userguide/kb-chunking.html?trk=article-ssr-frontend-pulse_little-text-block

✗ B Use Amazon ElastiCache to implement semantic caching for common queries. Use Anthropic Claude Sonnet to reformulate user queries. Configure real-time model feedback loops to continuously improve response quality based on user interactions.

Caching can help serve repeated queries, but it does not address the initial retrieval problem for new questions or ensure that relevant passages from the textbooks are retrieved. Query reformulation and feedback loops also do not specify how to preserve source context or filter low-relevance retrieved passages, which are the problems observed during testing.

✓ C Apply hierarchical chunking with both 200-token chunks and 1,000-token chunks. Implement hybrid search that combines vector and keyword search. Adjust the relevance score threshold to filter out low-confidence retrieval results.

Hierarchical chunking can provide a focused smaller chunk along with the larger context around it, addressing the risk of missing critical information from long textbooks. Hybrid search uses both semantic similarity and keyword matching, while a relevance score threshold can filter weak matches before they contribute to an answer. These choices directly target the irrelevant and incomplete retrieval results observed in testing. [8, 10, 11]

https://docs.aws.amazon.com/bedrock/latest/APIReference/API_agent_ChunkingConfiguration.html

<https://docs.aws.amazon.com/bedrock/latest/userguide/kb-test-config.html>

https://docs.aws.amazon.com/bedrock/latest/userguide/kb-chunking.html?trk=article-ssr-frontend-pulse_little-text-block

✗ D Use semantic chunking to automatically segment documents based on topic boundaries. Configure the knowledge base to use a single embedding model for all content types. Implement query expansion to reformulate user questions before retrieval.

Topic-based segmentation and query expansion may help retrieval, but this option does not include hybrid keyword search or a relevance threshold to screen out low-confidence results. It also does not explicitly retrieve both a focused chunk and its larger context, so it is less directly suited to the observed combination of irrelevant results and missing source context than option C. [4, 7, 11, 12]

<https://docs.aws.amazon.com/bedrock/latest/userguide/kb-managed-customize-ingestion.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/knowledge-base-setup.html>

https://docs.aws.amazon.com/bedrock/latest/userguide/kb-chunking.html?trk=article-ssr-frontend-pulse_little-text-block

<https://docs.aws.amazon.com/bedrock/latest/userguide/knowledge-base-generate-query.html>

Question 2

Multiple choice · Domain: D1 Foundation Model (FM) Integration, Data Management, and Compliance › Implement Retrieval Augmented Generation (RAG) solutions

An ecommerce company is using an Anthropic Claude Sonnet model in Amazon Bedrock to generate product recommendations. An AWS Lambda function retrieves customer purchase data from Amazon DynamoDB, product reviews from Amazon S3, and customer profile information from Amazon RDS.

Then the function sends the data directly to the Amazon Bedrock model through API calls. Recently, customers who have extensive purchase histories have begun to receive incomplete recommendations.

Amazon CloudWatch logs for the Lambda function show execution timeouts. CloudWatch logs for Amazon Bedrock API calls show intermittent errors. The company reviews the logs and finds that some requests are failing with context-length-exceeded errors. Other requests finish but appear to ignore portions of the input data.

The company wants the recommendation system to consider all customer data when the system generates recommendations. The company wants to use Amazon Bedrock Knowledge Bases to improve data organization and retrieval.

Which combination of solutions will meet these requirements? (Select TWO.)

- A.** Implement a chunking strategy that divides the customer data into smaller segments. Configure the model to process each segment separately. Invoke the model a final time to synthesize the individual responses into comprehensive recommendations.
- B.** Modify the prompt structure to place the most critical information at the beginning and end of the context window. Implement token-counting logic to truncate less important data when the interaction approaches the model's maximum context length.
- C.** Replace Claude Sonnet with a model that has a larger context-window capacity. Increase the Lambda function timeout to accommodate longer processing times for larger inputs.
- D.** Configure the recommendation system to use the Converse API. Modify the `additionalModelRequestFields` parameter to increase the maximum token limit beyond the model's default context-window size.
- E.** Implement RAG by using a knowledge base to index the customer data with vector embeddings. Retrieve only the most semantically relevant information for each recommendation request based on the current customer context.

Answer

- **A. Implement a chunking strategy that divides the customer data into smaller segments. Configure the model to process each segment separately. Invoke the model a final time to synthesize the individual responses into comprehensive recommendations.**
- **E. Implement RAG by using a knowledge base to index the customer data with vector embeddings. Retrieve only the most semantically relevant information for each recommendation request based on the current customer context.**

Explanation

The customer's large histories exceed the model's context capacity, and sending all data directly also contributes to timeouts and incomplete recommendations. Chunking the data and synthesizing the

segment-level results lets the model process information in manageable pieces. A Knowledge Base can index the customer data and retrieve relevant information for each request, reducing the amount of raw data sent while grounding recommendations in the customer context.

<https://docs.aws.amazon.com/bedrock/latest/userguide/kb-chunking.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/kb-how-retrieval.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/kb-how-it-works.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/knowledge-base.html>

✓ **A** Implement a chunking strategy that divides the customer data into smaller segments. Configure the model to process each segment separately. Invoke the model a final time to synthesize the individual responses into comprehensive recommendations.

Processing smaller segments avoids sending an oversized input that exceeds the model's context length. A final synthesis step can combine the segment-level results into a comprehensive recommendation instead of dropping portions of the customer data.

<https://docs.aws.amazon.com/bedrock/latest/userguide/kb-chunking.html>

✗ **B** Modify the prompt structure to place the most critical information at the beginning and end of the context window. Implement token-counting logic to truncate less important data when the interaction approaches the model's maximum context length.

Prioritizing information in the prompt does not ensure that all customer data is considered. Truncating data explicitly removes information, which conflicts with the requirement to use all customer data when generating recommendations.

✗ **C** Replace Claude Sonnet with a model that has a larger context-window capacity. Increase the Lambda function timeout to accommodate longer processing times for larger inputs.

A larger context window and longer Lambda timeout might accommodate more input, but this approach still sends large amounts of raw data directly to the model and does not use Knowledge Bases to organize and retrieve it. Increasing a timeout does not prevent context-length errors if the input still exceeds the model's limit.

<https://docs.aws.amazon.com/lambda/latest/dg/configuration-timeout.html>

✗ **D** Configure the recommendation system to use the Converse API. Modify the `additionalModelRequestFields` parameter to increase the maximum token limit beyond the model's default context-window size.

Using the Converse API does not let a caller raise a model's context-window capacity beyond its supported limit. This would not solve the oversized-input errors or provide the Knowledge Base retrieval requested in the scenario.

https://docs.aws.amazon.com/bedrock/latest/APIReference/API_runtime_Converse.html

✓ **E** Implement RAG by using a knowledge base to index the customer data with vector embeddings.

Retrieve only the most semantically relevant information for each recommendation request based on the current customer context.

A Knowledge Base can organize indexed data and retrieve relevant information for a given request, so the model receives useful customer context rather than the entire raw data set. This RAG approach helps address oversized prompts while grounding recommendations in the indexed customer data.

<https://docs.aws.amazon.com/bedrock/latest/userguide/kb-how-retrieval.html>

https://docs.aws.amazon.com/en_us/bedrock/latest/userguide/kb-how-data.html

<https://docs.aws.amazon.com/bedrock/latest/userguide/kb-how-it-works.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/knowledge-base.html>

https://docs.aws.amazon.com/en_en/bedrock/latest/userguide/kb-test-retrieve-generate.html

Question 3

Single choice · Domain: D1 Foundation Model (FM) Integration, Data Management, and Compliance › Implement data management solutions for GenAI applications

A research company is developing a GenAI system to produce summaries of technical documents. The company must catalog all data sources in a central location. The company needs a solution that can automatically discover and update data sources. The solution must tag each generated summary with citations as metadata that users can query. The solution must retain tamper-evident, immutable audit logs for every model invocation and store input/output records. Which solution will meet these requirements?

- A.** Use Amazon Comprehend to identify data sources in the documents. Store generated summaries in Amazon S3 and enable S3 Object Lock. Use Amazon CloudWatch metrics to generate reports about application throughput. Do not include logs for each invocation.
- B.** Use AWS Glue Data Catalog with crawlers to maintain data sources. Store generated summaries in Amazon S3. Write object tags that include a source ID. Store Amazon Bedrock model invocation logs in Amazon S3. Enable S3 Object Lock on the S3 bucket that stores invocation logs. Use AWS CloudTrail log file integrity validation to provide tamper-evident immutability.
- C.** Store application outputs in Amazon DynamoDB. Apply item-level tags that include source attribution. Write application events to Amazon CloudWatch Logs. Use IAM roles to provide audit traceability.
- D.** Use AWS AppConfig feature flags to implement data versioning. Restrict access to the model by using IAM condition keys. Maintain a versioned mapping file of source-to-output relationships in Amazon S3.

Answer: B. Use AWS Glue Data Catalog with crawlers to maintain data sources. Store generated summaries in Amazon S3. Write object tags that include a source ID. Store Amazon Bedrock model invocation logs in Amazon S3. Enable S3 Object Lock on the S3 bucket that stores invocation logs. Use AWS CloudTrail log file integrity validation to provide tamper-evident immutability.

Explanation

B meets the requirements by using AWS Glue Data Catalog crawlers to discover and keep catalog information about data sources current. It stores source identifiers with generated summaries as object tags, and Amazon Bedrock invocation logging records model inputs and outputs in S3. S3 Object Lock helps prevent the invocation-log records from being altered or deleted during the retention period, while CloudTrail log file integrity validation makes CloudTrail log files tamper-evident. [1][2][3][4]

<https://docs.aws.amazon.com/bedrock/latest/userguide/model-invocation-logging.html>

<https://docs.aws.amazon.com/glue/latest/dg/populate-catalog-methods.html>

<https://docs.aws.amazon.com/glue/latest/dg/catalog-and-crawler.html>

<https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudtrail-log-file-validation-digest-file-structure.html>

✗ A Use Amazon Comprehend to identify data sources in the documents. Store generated summaries in Amazon S3 and enable S3 Object Lock. Use Amazon CloudWatch metrics to generate reports about application throughput. Do not include logs for each invocation.

Comprehend and throughput metrics do not provide the required central catalog that automatically discovers and updates data sources. More importantly, this option explicitly omits per-invocation logs, so it cannot retain the required input/output records or provide an audit trail for every model invocation.

✓ B Use AWS Glue Data Catalog with crawlers to maintain data sources. Store generated summaries in Amazon S3. Write object tags that include a source ID. Store Amazon Bedrock model invocation logs in

Amazon S3. Enable S3 Object Lock on the S3 bucket that stores invocation logs. Use AWS CloudTrail log file integrity validation to provide tamper-evident immutability.

Glue crawlers populate and update the Data Catalog by discovering data sources. Source IDs in summary object tags provide source-attribution metadata, and Bedrock invocation logging captures model inputs and outputs in S3. Object Lock protects the stored invocation logs from alteration or deletion during retention, while CloudTrail log file integrity validation supports tamper-evident validation of CloudTrail log files.

<https://docs.aws.amazon.com/bedrock/latest/userguide/model-invocation-logging.html>

<https://docs.aws.amazon.com/glue/latest/dg/populate-catalog-methods.html>

<https://docs.aws.amazon.com/glue/latest/dg/catalog-and-crawler.html>

<https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudtrail-log-file-validation-digest-file-structure.html>

<https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudtrail-log-file-validation-enabling.html>

✗ C Store application outputs in Amazon DynamoDB. Apply item-level tags that include source attribution. Write application events to Amazon CloudWatch Logs. Use IAM roles to provide audit traceability. This option does not provide a central, automatically discovered and updated data-source catalog such as Glue Data Catalog crawlers. CloudWatch application events and IAM roles also do not establish the specified immutable, tamper-evident storage of per-invocation input/output records.

✗ D Use AWS AppConfig feature flags to implement data versioning. Restrict access to the model by using IAM condition keys. Maintain a versioned mapping file of source-to-output relationships in Amazon S3. A versioned mapping file is not an automatically maintained central catalog of discovered data sources. The option also does not specify per-invocation input/output logging or tamper-evident immutable audit-log retention.

Question 4

Multiple choice · Domain: D1 Foundation Model (FM) Integration, Data Management, and Compliance › Implement prompt engineering techniques

A university is building an AI-powered application that includes several sub-applications. The subapplications include AI assistants, assignment graders, and internal analytics applications. The university is defining and testing multiple prompts by using various foundation models (FMs). The university wants to compare variants of each prompt and choose the variant that yield outputs that are best-suited for specified use cases. The university requires a version control solution for the prompts. The university must be able to test prompt variations and collect audit trails for prompt changes and usage. The solution must also maintain consistency while allowing the prompts to integrate into the main application. Which combination of solutions will meet these requirements with the LEAST operational overhead? (Select TWO.)

- A.** Use Amazon Bedrock Prompt Management to create versioned prompts. Include parameterized variables for each use case.
- B.** Store prompts in Amazon S3. Use AWS Step Functions to orchestrate the model interactions and service integrations.
- C.** Use Amazon Bedrock Flows to create workflows that combine FMs and AWS services.
- D.** Configure AWS Config to record prompt changes. Use AWS CloudTrail to track prompt usage.
- E.** Configure Amazon Bedrock intelligent prompt routing.

Answer

- A. Use Amazon Bedrock Prompt Management to create versioned prompts. Include parameterized variables for each use case.
- C. Use Amazon Bedrock Flows to create workflows that combine FMs and AWS services.

Explanation

Amazon Bedrock Prompt Management provides a purpose-built way to define prompts with variables, create versions and variants, and test them with foundation models, helping the university compare outputs across use cases. Amazon Bedrock Flows can combine those prompts with models and AWS services in reusable workflows that integrate with the application. Together, these managed capabilities address prompt iteration and application integration without requiring the university to build and operate a separate prompt repository and orchestration layer.

<https://docs.aws.amazon.com/bedrock/latest/userguide/prompt-management.html>

https://docs.aws.amazon.com/us_en/bedrock/latest/userguide/flows.html

<https://docs.aws.amazon.com/bedrock/latest/userguide/prompt-management-create.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/flows-ex-prompt.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/flows-create.html>

✓ **A** Use Amazon Bedrock Prompt Management to create versioned prompts. Include parameterized variables for each use case.

Prompt Management supports creating prompts with variables and maintaining versions, so the university can reuse prompts across use cases and track changes. It also supports prompt variants and testing, which helps compare results from different prompt designs and foundation models.

<https://docs.aws.amazon.com/bedrock/latest/userguide/prompt-management.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/prompt-management-create.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/advanced-prompt-optimization-input.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/prompt-management-optimize.html>

✗ **B** Store prompts in Amazon S3. Use AWS Step Functions to orchestrate the model interactions and service integrations.

This approach would require the university to build and maintain its own prompt versioning, variant comparison, and audit processes around S3 and Step Functions. It adds operational work rather than using Bedrock's purpose-built prompt and flow features.

<https://docs.aws.amazon.com/step-functions/latest/dg/getting-started.html>

<https://docs.aws.amazon.com/step-functions/latest/dg/supported-services-awssdk.html>

✓ **C** Use Amazon Bedrock Flows to create workflows that combine FMs and AWS services.

Bedrock Flows provides a managed way to connect prompts, foundation models, and AWS services into workflows that can be invoked by the application. This supports consistent, reusable integration across the university's sub-applications while avoiding a separately operated orchestration layer.

https://docs.aws.amazon.com/us_en/bedrock/latest/userguide/flows.html

<https://docs.aws.amazon.com/bedrock/latest/userguide/flows-ex-prompt.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/flows-create.html>

✗ **D** Configure AWS Config to record prompt changes. Use AWS CloudTrail to track prompt usage.

Config and CloudTrail can provide general configuration and API activity records, but they do not provide the prompt-variant creation and comparison workflow the university needs. Using them as the solution would also require extra setup and would not itself integrate prompts into the application.

✗ **E** Configure Amazon Bedrock intelligent prompt routing.

Prompt routing selects an appropriate model for a prompt; it does not manage prompt versions or compare prompt variants. It therefore does not meet the university's requirements for prompt change control and testing.

<https://docs.aws.amazon.com/bedrock/latest/userguide/prompt-routing.html>

Question 5

Single choice · Domain: D1 Foundation Model (FM) Integration, Data Management, and Compliance › Implement data management solutions for GenAI applications

A research company is developing a GenAI system to produce summaries of technical documents.

The company must catalog all data sources in a central location. The company needs a solution that can automatically discover and update data sources. The solution must tag each generated summary with citations as metadata that users can query. The solution must retain tamper-evident, immutable audit logs for every model invocation and store I/O records. Which solution will meet these requirements?

- A.** Use Amazon Comprehend to identify data sources in the documents. Store generated summaries in Amazon S3 and enable S3 Object Lock. Use Amazon CloudWatch metrics to generate reports about application throughput. Do not include logs for each invocation.
- B.** Use AWS Glue Data Catalog with crawlers to maintain data sources. Store generated summaries in Amazon S3. Write object tags that include a source ID. Store Amazon Bedrock model invocation logs in Amazon S3. Enable S3 Object Lock on the S3 bucket that stores invocation logs. Use AWS CloudTrail log file integrity validation to provide tamper-evident immutability.
- C.** Store application outputs in Amazon DynamoDB. Apply item-level tags that include source attribution. Write application events to Amazon CloudWatch Logs. Use IAM roles to provide audit traceability.
- D.** Use AWS AppConfig feature flags to implement data versioning. Restrict access to the model by using IAM condition keys. Maintain a versioned mapping file of source-to-output relationships in Amazon S3.

Answer: B. Use AWS Glue Data Catalog with crawlers to maintain data sources. Store generated summaries in Amazon S3. Write object tags that include a source ID. Store Amazon Bedrock model invocation logs in Amazon S3. Enable S3 Object Lock on the S3 bucket that stores invocation logs. Use AWS CloudTrail log file integrity validation to provide tamper-evident immutability.

Explanation

AWS Glue Data Catalog with crawlers provides a central catalog that can discover data sources and update catalog information as sources change. Storing a source ID in tags on each generated summary gives users source-attribution metadata to query. Amazon Bedrock model invocation logging can send invocation records to Amazon S3, while S3 Object Lock retains those records immutably and CloudTrail log file integrity validation helps detect changes to CloudTrail log files. Together, these components address the cataloging, citation, and audit requirements.

<https://docs.aws.amazon.com/bedrock/latest/userguide/model-invocation-logging.html>

<https://docs.aws.amazon.com/glue/latest/dg/catalog-and-crawler.html>

<https://docs.aws.amazon.com/glue/latest/dg/add-crawler.html>

https://docs.aws.amazon.com/en_en/awsccloudtrail/latest/userguide/cloudtrail-log-file-validation-intro.html

✗ **A** Use Amazon Comprehend to identify data sources in the documents. Store generated summaries in Amazon S3 and enable S3 Object Lock. Use Amazon CloudWatch metrics to generate reports about application throughput. Do not include logs for each invocation.

Comprehend and S3 Object Lock do not provide the required central, automatically maintained data-source catalog or queryable citation tags. Throughput metrics are not records of each model invocation, and the option explicitly omits those logs.

✓ **B** Use AWS Glue Data Catalog with crawlers to maintain data sources. Store generated summaries in Amazon S3. Write object tags that include a source ID. Store Amazon Bedrock model invocation logs in

Amazon S3. Enable S3 Object Lock on the S3 bucket that stores invocation logs. Use AWS CloudTrail log file integrity validation to provide tamper-evident immutability.

Glue crawlers populate and update the Glue Data Catalog, meeting the central discovery and maintenance requirement. A source ID in summary object tags provides queryable source attribution, while Bedrock invocation logging records model activity in S3. Object Lock supports immutable retention of those records, and CloudTrail log file integrity validation helps detect tampering with CloudTrail log files.

<https://docs.aws.amazon.com/bedrock/latest/userguide/model-invocation-logging.html>

<https://docs.aws.amazon.com/glue/latest/dg/catalog-and-crawler.html>

<https://docs.aws.amazon.com/glue/latest/dg/add-crawler.html>

https://docs.aws.amazon.com/en_en/awsccloudtrail/latest/userguide/cloudtrail-log-file-validation-intro.html

X C Store application outputs in Amazon DynamoDB. Apply item-level tags that include source attribution. Write application events to Amazon CloudWatch Logs. Use IAM roles to provide audit traceability.

This option does not provide the required automatically discovered and updated central catalog of data sources. Application events and IAM roles also do not, as described, provide tamper-evident immutable logs for every model invocation and store I/O record.

X D Use AWS AppConfig feature flags to implement data versioning. Restrict access to the model by using IAM condition keys. Maintain a versioned mapping file of source-to-output relationships in Amazon S3. A versioned mapping file is not an automatically discovered and updated central data catalog, and the option does not describe queryable citation metadata on each summary. It also lacks per-invocation and store I/O logs with immutable, tamper-evident retention.

Question 6

Single choice · Domain: D1 Foundation Model (FM) Integration, Data Management, and Compliance › Implement Retrieval Augmented Generation (RAG) solutions

An ecommerce company is developing a generative AI application that uses Amazon Bedrock with Anthropic Claude to recommend products to customers. Customers report that some recommended products are not available for sale on the website or are not relevant to the customer. Customers also report that the solution takes a long time to generate some recommendations.

The company investigates the issues and finds that most interactions between customers and the product recommendation solution are unique. The company confirms that the solution recommends products that are not in the company's product catalog. The company must resolve these issues.

Which solution will meet this requirement?

- A.** Increase grounding within Amazon Bedrock Guardrails. Enable Automated Reasoning checks. Set up provisioned throughput.
- B.** Use prompt engineering to restrict the model responses to relevant products. Use streaming techniques such as the `InvokeModelWithResponseStream` action to reduce perceived latency for the customers.
- C.** Create an Amazon Bedrock knowledge base. Implement Retrieval Augmented Generation RAG. Set the `PerformanceConfigLatency` parameter to optimized.
- D.** Store product catalog data in Amazon OpenSearch Service. Validate the model's product recommendations against the product catalog. Use Amazon DynamoDB to implement response caching.

Answer: C. Create an Amazon Bedrock knowledge base. Implement Retrieval Augmented Generation RAG. Set the `PerformanceConfigLatency` parameter to optimized.

Dumpsgate

Question 7

Single choice · Domain: D1 Foundation Model (FM) Integration, Data Management, and Compliance › Implement Retrieval Augmented Generation (RAG) solutions

A book publishing company wants to build a book recommendation system that uses an AI assistant. The AI assistant will use ML to generate a list of recommended books from the company's book catalog. The system must suggest books based on conversations with customers.

The company stores the text of the books, customers' and editors' reviews of the books, and extracted book metadata in Amazon S3. The system must support low-latency responses and scale efficiently to handle more than 10,000 concurrent users.

Which solution will meet these requirements?

- A.** Use Amazon Bedrock Knowledge Bases to generate embeddings. Store the embeddings as a vector store in Amazon OpenSearch Service. Create an AWS Lambda function that queries the knowledge base. Configure Amazon API Gateway to invoke the Lambda function when handling user requests.
- B.** Use Amazon Bedrock Knowledge Bases to generate embeddings. Store the embeddings as a vector store in Amazon DynamoDB. Create an AWS Lambda function that queries the knowledge base. Configure Amazon API Gateway to invoke the Lambda function when handling user requests.
- C.** Use Amazon SageMaker AI to deploy a pre-trained model to build a personalized recommendation engine for books. Deploy the model as a SageMaker AI endpoint. Invoke the model endpoint by using Amazon API Gateway.
- D.** Create an Amazon Kendra GenAI Enterprise Edition index that uses the S3 connector to index the book catalog data stored in Amazon S3. Configure built-in FAQ in the Kendra index. Develop an AWS Lambda function that queries the Kendra index based on user conversations. Deploy Amazon API Gateway to expose this functionality and invoke the Lambda function.

Answer: A. Use Amazon Bedrock Knowledge Bases to generate embeddings. Store the embeddings as a vector store in Amazon OpenSearch Service. Create an AWS Lambda function that queries the knowledge base. Configure Amazon API Gateway to invoke the Lambda function when handling user requests.

Explanation

Option A uses Amazon Bedrock Knowledge Bases to create embeddings from the book catalog data in Amazon S3 and OpenSearch Service as the vector store for semantic retrieval. That lets the assistant use a customer's conversation to find relevant books, reviews, and metadata, while the Lambda and API Gateway path exposes the retrieval capability to users. OpenSearch is a supported vector store for Bedrock Knowledge Bases and is a better fit here than DynamoDB for vector retrieval.

https://docs.aws.amazon.com/en_en/bedrock/latest/userguide/knowledge-base-supported.html

<https://docs.aws.amazon.com/bedrock/latest/userguide/knowledge-base-build.html>

https://docs.aws.amazon.com/en_us/bedrock/latest/userguide/kb-how-data.html

<https://docs.aws.amazon.com/pdfs/prescriptive-guidance/latest/retrieval-augmented-generation-options/retrieval-augmented-generation-options.pdf>

✓ **A** Use Amazon Bedrock Knowledge Bases to generate embeddings. Store the embeddings as a vector store in Amazon OpenSearch Service. Create an AWS Lambda function that queries the knowledge base. Configure Amazon API Gateway to invoke the Lambda function when handling user requests. Bedrock Knowledge Bases can ingest the company's S3 content and generate embeddings for semantic retrieval, so the assistant can use conversational context to find relevant books. OpenSearch Service is a

supported vector store for Knowledge Bases, and the Lambda/API Gateway path provides an interface for customer requests. This architecture fits the requested conversational recommendations better than an FAQ lookup.

https://docs.aws.amazon.com/en_en/bedrock/latest/userguide/knowledge-base-supported.html

<https://docs.aws.amazon.com/bedrock/latest/userguide/knowledge-base-build.html>

https://docs.aws.amazon.com/en_us/bedrock/latest/userguide/kb-how-data.html

<https://docs.aws.amazon.com/pdfs/prescriptive-guidance/latest/retrieval-augmented-generation-options/retrieval-augmented-generation-options.pdf>

✗ B Use Amazon Bedrock Knowledge Bases to generate embeddings. Store the embeddings as a vector store in Amazon DynamoDB. Create an AWS Lambda function that queries the knowledge base. Configure Amazon API Gateway to invoke the Lambda function when handling user requests. The overall Knowledge Bases and API pattern is suitable, but DynamoDB is not a supported vector-store choice for Bedrock Knowledge Bases in the documented supported-store options. Use a supported vector store, such as OpenSearch Service, for the embeddings and semantic retrieval required here.

https://docs.aws.amazon.com/en_en/bedrock/latest/userguide/knowledge-base-supported.html

✗ C Use Amazon SageMaker AI to deploy a pre-trained model to build a personalized recommendation engine for books. Deploy the model as a SageMaker AI endpoint. Invoke the model endpoint by using Amazon API Gateway.

Deploying a model endpoint does not by itself make use of the company's S3 book text, reviews, and metadata or provide conversational retrieval over that content. The option also does not describe the data preparation or recommendation model needed to produce personalized recommendations from those sources.

<https://docs.aws.amazon.com/sagemaker/latest/dg/instance-recommendation-create.html>

✗ D Create an Amazon Kendra GenAI Enterprise Edition index that uses the S3 connector to index the book catalog data stored in Amazon S3. Configure built-in FAQ in the Kendra index. Develop an AWS Lambda function that queries the Kendra index based on user conversations. Deploy Amazon API Gateway to expose this functionality and invoke the Lambda function.

Kendra can index content from S3 and support search, but configuring built-in FAQ is aimed at answering predefined questions and is not the described vector-based approach for finding recommended books from conversational context. The option does not establish the book-recommendation behavior required by the scenario.

<https://docs.aws.amazon.com/kendra/latest/dg/in-creating-faq.html>

<https://docs.aws.amazon.com/kendra/latest/dg/data-source-s3.html>

<https://docs.aws.amazon.com/kendra/latest/dg/how-it-works.html>

Question 8

Single choice · Domain: D1 Foundation Model (FM) Integration, Data Management, and Compliance › Implement Retrieval Augmented Generation (RAG) solutions

A healthcare company is developing an application to process medical queries. The application must answer complex queries with high accuracy by reducing semantic dilution. The application must refer to domain-specific terminology in medical documents to reduce ambiguity in medical terminology. The application must be able to respond to 1,000 queries each minute with response times less than 2 seconds.

Which solution will meet these requirements with the LEAST operational overhead?

- A.** Use Amazon API Gateway to route incoming queries to an Amazon Bedrock agent. Configure the agent to use an Anthropic Claude model to decompose queries and an Amazon Titan model to expand queries. Create an Amazon Bedrock knowledge base to store the reference medical documents.
- B.** Configure an Amazon Bedrock knowledge base to store the reference medical documents. Enable query decomposition in the knowledge base. Configure an Amazon Bedrock flow that uses a foundation model and the knowledge base to support the application.
- C.** Use Amazon SageMaker AI to host custom ML models for both query decomposition and query expansion. Configure Amazon Bedrock knowledge bases to store the reference medical documents. Encrypt the documents in the knowledge base.
- D.** Create an Amazon Bedrock agent to orchestrate multiple AWS Lambda functions to decompose queries. Create an Amazon Bedrock knowledge base to store the reference medical documents. Use the agent's built-in knowledge base capabilities. Add deep research and reasoning capabilities to the agent to reduce ambiguity in the medical terminology.

Answer: B. Configure an Amazon Bedrock knowledge base to store the reference medical documents. Enable query decomposition in the knowledge base. Configure an Amazon Bedrock flow that uses a foundation model and the knowledge base to support the application.

Explanation

B uses a Bedrock knowledge base with query decomposition to retrieve medical-document context for complex queries, helping keep retrieval focused on domain-specific terminology. A Bedrock flow can connect the foundation model and knowledge base in a managed workflow, avoiding the custom model hosting and additional orchestration components in the other choices. This is the best fit for the low-operational-overhead requirement; the stated query rate and latency would still need to be validated for the deployed configuration.

https://docs.aws.amazon.com/us_en/bedrock/latest/userguide/flows.html

<https://docs.aws.amazon.com/bedrock/latest/userguide/kb-test-config.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/flows-how-it-works.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/kb-how-it-works.html>

X A Use API Gateway to route queries to a Bedrock agent, with Claude for query decomposition, Titan for query expansion, and a Bedrock knowledge base.

This adds an agent and multiple model steps rather than using the knowledge base's query-decomposition configuration with a flow. The extra orchestration and query expansion are not the lowest-overhead approach for this scenario, and expansion may add terms beyond the focused medical context the application needs.

<https://docs.aws.amazon.com/bedrock/latest/userguide/kb-test-config.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/kb-how-it-works.html>

✓ **B** Configure a Bedrock knowledge base with the medical documents and query decomposition, then use a Bedrock flow with a foundation model and the knowledge base.

Query decomposition lets retrieval handle a complex query as focused parts, while the knowledge base supplies relevant context from the medical documents, including domain terminology. Bedrock flows provide a managed way to connect the foundation model and knowledge base, making this the best fit among the options for accuracy with less operational overhead.

https://docs.aws.amazon.com/us_en/bedrock/latest/userguide/flows.html

<https://docs.aws.amazon.com/bedrock/latest/userguide/kb-test-config.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/flows-how-it-works.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/kb-how-it-works.html>

✗ **C** Host custom query-decomposition and query-expansion models on SageMaker AI, use Bedrock knowledge bases, and encrypt the documents.

Hosting custom models on SageMaker adds model deployment and maintenance work that the scenario's least-operational-overhead requirement argues against. Encryption protects data but does not itself improve query accuracy, reduce semantic dilution, or address response time.

<https://docs.aws.amazon.com/bedrock/latest/userguide/kb-how-it-works.html>

✗ **D** Use a Bedrock agent to orchestrate multiple Lambda functions for query decomposition, a knowledge base, and deep research and reasoning capabilities.

Orchestrating multiple Lambda functions and adding deep research and reasoning introduces more components than the managed flow approach. Those additional steps are not needed to use document-grounded retrieval and could make the low-latency requirement harder to meet.

https://docs.aws.amazon.com/us_en/bedrock/latest/userguide/flows.html

<https://docs.aws.amazon.com/bedrock/latest/userguide/flows-how-it-works.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/kb-how-it-works.html>

Question 9

Single choice · Domain: D1 Foundation Model (FM) Integration, Data Management, and Compliance › Implement Retrieval Augmented Generation (RAG) solutions

A financial services company wants to develop an Amazon Bedrock application that gives analysts the ability to query quarterly earnings reports and financial statements. The financial documents are typically 5–100 pages long and contain both tabular data and text. The application must provide contextually accurate responses that preserve the relationship between financial metrics and their explanatory text. To support accurate and scalable retrieval, the application must incorporate document segmentation and context management strategies.

Which solution will meet these requirements?

- A.** Use a direct model invocation approach that uses Anthropic Claude to process each financial document as a single input. Use fine-tuned prompts that instruct the model to parse tables and text separately.
- B.** Use Amazon Bedrock Knowledge Bases to create a Retrieval Augmented Generation (RAG) application that retrieves relevant information from contextually chunked sections of financial documents. Segment documents based on their structural layout. Include citations that reference the original source materials.
- C.** Deploy an Amazon Bedrock agent that has an action group that calls custom AWS Lambda functions to analyze financial documents. Configure the Lambda functions to perform fixed-size chunking when a user submits a query about financial metrics.
- D.** Create one specialized Amazon Bedrock application that is optimized for structured data. Create a second application that is optimized for unstructured data. Configure each application to use a tailored chunking strategy that is suited to the application's content type. Implement logic to link queries to the appropriate sources.

Answer: B. Use Amazon Bedrock Knowledge Bases to create a Retrieval Augmented Generation (RAG) application that retrieves relevant information from contextually chunked sections of financial documents. Segment documents based on their structural layout. Include citations that reference the original source materials.

Explanation

Amazon Bedrock Knowledge Bases support a RAG workflow that retrieves relevant source content for a user's question, rather than requiring each long report to be processed as one input. Structural or contextual chunking helps keep financial metrics connected to the explanatory text around them, which is important for accurate answers from documents containing both tables and prose. Citing the retrieved source material also lets analysts trace answers back to the original reports. This makes option B the best fit for accurate, scalable retrieval.

<https://docs.aws.amazon.com/bedrock/latest/userguide/kb-data-source-customize-ingestion.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/kb-how-retrieval.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/knowledge-base.html>

✗ A Use a direct model invocation approach that uses Anthropic Claude to process each financial document as a single input. Use fine-tuned prompts that instruct the model to parse tables and text separately.

Processing each long report as one input does not provide a retrieval strategy for finding the relevant sections as the document collection grows. Asking the model to parse tables and text separately also does not ensure that a metric remains connected to its explanatory text or provide source citations.

✓ **B** Use Amazon Bedrock Knowledge Bases to create a Retrieval Augmented Generation (RAG) application that retrieves relevant information from contextually chunked sections of financial documents. Segment documents based on their structural layout. Include citations that reference the original source materials.

A Knowledge Base RAG workflow retrieves relevant portions of source documents to provide context for a response, making it suitable for querying long reports. Segmenting based on document structure helps preserve relationships between financial data and its accompanying explanations, while citations point analysts to the original material.

<https://docs.aws.amazon.com/bedrock/latest/userguide/kb-data-source-customize-ingestion.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/kb-how-retrieval.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/knowledge-base.html>

✗ **C** Deploy an Amazon Bedrock agent that has an action group that calls custom AWS Lambda functions to analyze financial documents. Configure the Lambda functions to perform fixed-size chunking when a user submits a query about financial metrics.

Fixed-size chunking can split a table or a financial metric from the text that explains it, which conflicts with the requirement to preserve their relationship. An agent action group and Lambda functions do not, by themselves, provide the contextually structured retrieval approach specified in the question.

<https://docs.aws.amazon.com/bedrock/latest/userguide/agents-lambda.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/agents-action-create.html>

https://docs.aws.amazon.com/bedrock/latest/APIReference/API_agent_ChunkingConfiguration.html

✗ **D** Create one specialized Amazon Bedrock application that is optimized for structured data. Create a second application that is optimized for unstructured data. Configure each application to use a tailored chunking strategy that is suited to the application's content type. Implement logic to link queries to the appropriate sources.

Separating tables from explanatory prose into different applications risks losing the very relationship the solution must preserve. It also adds routing and linking logic instead of retrieving the relevant contextual sections together from the source documents.

Question 10

Single choice · Domain: D1 Foundation Model (FM) Integration, Data Management, and Compliance › Implement prompt engineering techniques

A retail company runs an application that makes product recommendations to customers on the company's website. The application uses Amazon Bedrock to generate recommendations by dynamically constructing prompts and sending them to foundation models (FMs). A GenAI developer has deployed an update to the application that instructs the FM to include a specific promotional message when the FM generates a response to prompts. When the developer tests the application, the promotional message does not always appear in the responses. When the promotional message does appear in the responses, it does not always flow with the rest of the text. The GenAI developer must ensure that the promotional message always appears in the FM responses. Which solution will meet this requirement?

- A. Use an Amazon Bedrock Guardrails filter on the prompt. Set the input filter strength to HIGH.
- B. Generate multiple response variants that include the promotional message in different ways. Use a reranker model to select the most coherent version based on relevance to the original prompt.
- C. Run the prompt through Amazon Bedrock. Process the response through Amazon Bedrock AgentCore to add the promotional message. Rerank the results by using the original prompt and the desired message as context.
- D. Reinforce the requirement to include the new promotional message within product recommendations by using an output indicator in prompts to the FM.

Answer: D. Reinforce the requirement to include the new promotional message within product recommendations by using an output indicator in prompts to the FM.

Explanation

The requirement is that every generated recommendation include the promotional message, so the prompt should make that instruction explicit and reinforce it with an output indicator. This improves the likelihood that the FM follows the requested output structure and includes the message. Guardrails and reranking address different concerns and do not ensure that the required text appears in every response.

<https://docs.aws.amazon.com/bedrock/latest/userguide/design-a-prompt.html>

✗ **A** Use an Amazon Bedrock Guardrails filter on the prompt. Set the input filter strength to HIGH. Guardrails filters are used to detect or block specified input or output content, not to require the FM to add a promotional message. Increasing the input filter strength therefore does not ensure that the message appears in generated recommendations.

https://docs.aws.amazon.com/us_en/bedrock/latest/userguide/guardrails.html

https://docs.aws.amazon.com/en_en/bedrock/latest/userguide/guardrails-content-filters-overview.html

<https://docs.aws.amazon.com/bedrock/latest/userguide/guardrails-how.html>

✗ **B** Generate multiple response variants that include the promotional message in different ways. Use a reranker model to select the most coherent version based on relevance to the original prompt. A reranker can rank candidate results for relevance, but selecting the most relevant or coherent variant does not guarantee that the selected response includes the required message. The approach also depends on candidate generation already producing suitable variants.

<https://docs.aws.amazon.com/bedrock/latest/userguide/rerank.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/rerank-use.html>

✗ **C** Run the prompt through Amazon Bedrock. Process the response through Amazon Bedrock AgentCore to add the promotional message. Rerank the results by using the original prompt and the desired message as context.

AgentCore is not presented here as a mechanism for reliably inserting required text into an FM response, and reranking does not enforce inclusion of that text. This adds processing steps rather than using the prompt technique called for in this scenario.

https://docs.aws.amazon.com/bedrock-agentcore/latest/devguide/what-is-bedrock-agentcore.html?sc_channel=el
https://aws.amazon.com/bedrock/agentcore/faqs/?sc_channel=ps&trk=96c7de5d-e87e-4f2d-9f26-80ea70660d09
<https://aws.amazon.com/bedrock/agentcore/resources/>
<https://docs.aws.amazon.com/bedrock/latest/userguide/rerank.html>

✓ **D** Reinforce the requirement to include the new promotional message within product recommendations by using an output indicator in prompts to the FM.

An output indicator can clarify the expected response structure and reinforce that the recommendation must contain the promotional message. This directly addresses inconsistent inclusion and placement by making the desired output explicit in the prompt.

<https://docs.aws.amazon.com/bedrock/latest/userguide/design-a-prompt.html>

Question 11

Single choice · Domain: D2 Implementation and Integration › Implement deployment strategies for GenAI applications

A financial services company is building a fraud detection system by using Amazon Bedrock. The system will monitor activity in multiple stock trading applications that run in the United States and Europe. The system must process 1,000 transactions every second with sub-500 ms response times.

The system must also maintain high availability during connectivity disruptions.

The company must ensure that data for European customers is processed only in AWS Regions that are based in Europe.

Which solution will meet these requirements?

- A.** Configure AWS Lambda functions and Amazon EKS applications to use the InvokeModel API with a global inference profile. Deploy an automated failover system that uses Amazon Route 53 health checks. Create a dedicated European inference profile and enable geographic cross-Region inference for European applications. Use Amazon CloudWatch alarms to monitor utilization metrics.
- B.** Configure all applications to use the InvokeModel API with provisioned throughput for an Anthropic Claude model in each Region separately. Set up a custom Application Load Balancer to distribute traffic based on Regional capacity and response times. Implement a Regional failover mechanism that uses Amazon EventBridge rules to handle connectivity disruptions.
- C.** Configure all applications to use the InvokeModelWithResponseStream API with on-demand throughput. Deploy an Amazon API Gateway REST API with Regional endpoints in each location where the company operates to route requests to the closest Amazon Bedrock endpoint. Create separate IAM roles for applications that run in the United States and Europe. Grant the IAM roles Region-specific permissions.
- D.** Configure applications that run in the United States to use provisioned throughput with the InvokeModel API. Configure European applications to use a Europe-specific geographic inference profile to ensure data sovereignty. Configure automatic scaling for provisioned capacity based on utilization metrics. Use Amazon EventBridge and AWS Lambda functions to implement cross-Region failover mechanisms.

Answer: D. Configure applications that run in the United States to use provisioned throughput with the InvokeModel API. Configure European applications to use a Europe-specific geographic inference profile to ensure data sovereignty. Configure automatic scaling for provisioned capacity based on utilization metrics. Use Amazon EventBridge and AWS Lambda functions to implement cross-Region failover mechanisms.

Explanation

D separates the U.S. and European workloads so European customer data can use a Europe-specific geographic inference profile, which constrains cross-Region inference to the required geography. Provisioned throughput is the suitable choice for the stated sustained, high transaction volume, with utilization-based scaling helping adapt capacity. EventBridge and Lambda provide a mechanism to respond to connectivity disruptions and initiate cross-Region failover, while keeping European failover within Europe.

<https://docs.aws.amazon.com/bedrock/latest/userguide/scaling-throughput-best-practices.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/global-cross-region-inference.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/inference-profiles-use.html>

✗ **A** Configure AWS Lambda functions and Amazon EKS applications to use the InvokeModel API with a global inference profile. Deploy an automated failover system that uses Amazon Route 53 health checks. Create a dedicated European inference profile and enable geographic cross-Region inference for European applications. Use Amazon CloudWatch alarms to monitor utilization metrics.

A global inference profile can route inference across Regions beyond Europe, so using it for the applications conflicts with the requirement to keep European customer data in European Regions. Adding a European profile for European applications does not resolve that conflict if those applications are also configured to use the global profile.

<https://docs.aws.amazon.com/bedrock/latest/userguide/global-cross-region-inference.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/inference-profiles-use.html>

✗ **B** Configure all applications to use the InvokeModel API with provisioned throughput for an Anthropic Claude model in each Region separately. Set up a custom Application Load Balancer to distribute traffic based on Regional capacity and response times. Implement a Regional failover mechanism that uses Amazon EventBridge rules to handle connectivity disruptions.

Provisioned throughput can address high, predictable demand, but this option does not constrain European requests or failover to Regions in Europe. A capacity- or latency-based routing and failover setup could therefore send European customer data to a Region outside Europe.

<https://docs.aws.amazon.com/bedrock/latest/userguide/scaling-throughput-best-practices.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/global-cross-region-inference.html>

✗ **C** Configure all applications to use the InvokeModelWithResponseStream API with on-demand throughput. Deploy an Amazon API Gateway REST API with Regional endpoints in each location where the company operates to route requests to the closest Amazon Bedrock endpoint. Create separate IAM roles for applications that run in the United States and Europe. Grant the IAM roles Region-specific permissions.

On-demand throughput does not reserve capacity for the sustained 1,000-transactions-per-second workload, and response streaming by itself does not ensure the required throughput or response time. Routing to the closest endpoint also does not establish the explicit Europe-only geographic boundary needed for European customer data, and the option does not describe a connectivity-disruption failover mechanism.

<https://docs.aws.amazon.com/bedrock/latest/userguide/scaling-throughput-best-practices.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/global-cross-region-inference.html>

✓ **D** Configure applications that run in the United States to use provisioned throughput with the InvokeModel API. Configure European applications to use a Europe-specific geographic inference profile to ensure data sovereignty. Configure automatic scaling for provisioned capacity based on utilization metrics. Use Amazon EventBridge and AWS Lambda functions to implement cross-Region failover

mechanisms.

Provisioned throughput with utilization-based scaling is suited to the high transaction volume, while the Europe-specific geographic inference profile keeps European inference within the permitted geography. EventBridge and Lambda can coordinate failover during connectivity disruptions; for the sovereignty requirement, European failover must remain within Europe.

<https://docs.aws.amazon.com/bedrock/latest/userguide/scaling-throughput-best-practices.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/global-cross-region-inference.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/inference-profiles-use.html>

Question 12

Single choice · Domain: D2 Implementation and Integration › Implement APIs and SDKs for GenAI applications

A company uses an application to process customer support tickets. The company wants to integrate AI-powered sentiment analysis and auto-response generation into the application by using Amazon Bedrock. The company wants to prioritize urgent issues and reduce initial response times by 40% compared to manual responses. The solution must process 100 concurrent webhook requests with response times under 500 ms. The solution must maintain 99.9% availability across multiple AWS Regions and authenticate all incoming requests. The company must avoid any authentication failures. The company does not want to modify the existing application infrastructure, which includes several ticketing systems that use multiple webhook authentication methods. The solution must support scaling to handle occasional spikes up to 250,000 daily tickets during peak periods. Which solution will meet these requirements?

- A.** Use an Amazon API Gateway REST API with a Regional endpoint to receive webhook requests and invoke AWS Lambda functions. Configure Lambda authorizers to validate all the webhook authentication methods. Configure the Lambda functions to call Amazon Bedrock to perform sentiment analysis and generate responses. Store results in Amazon DynamoDB global tables to provide multi-Region availability.
- B.** Create AWS Lambda function URLs for each ticketing system. Configure the function URLs with the NONE authentication type. Configure separate Lambda functions to verify webhook signatures by using Hash-based Message Authentication Code (HMAC) validation in the function code. Deploy the functions to multiple Regions and use AWS Global Accelerator to route traffic. Use Amazon Bedrock to perform sentiment analysis and generate responses. Return responses through webhook callbacks.
- C.** Set up an Amazon SQS queue in each Region to receive webhook messages. Use the SQS queue to invoke AWS Lambda functions that call Amazon Comprehend to perform sentiment analysis and Amazon Lex to generate responses. Use Amazon EventBridge to retry message delivery to the application API.
- D.** Deploy an AWS AppSync GraphQL API to multiple Regions. Configure API tokens to authenticate incoming requests. Create GraphQL mutation resolvers that publish events to Amazon EventBridge. Configure EventBridge rules to invoke AWS Lambda functions that use Amazon Bedrock to perform sentiment analysis and generate responses. Use Amazon CloudFront to reduce latency.

Answer: A. Use an Amazon API Gateway REST API with a Regional endpoint to receive webhook requests and invoke AWS Lambda functions. Configure Lambda authorizers to validate all the webhook authentication methods. Configure the Lambda functions to call Amazon Bedrock to perform sentiment analysis and generate responses. Store results in Amazon DynamoDB global tables to provide multi-Region availability.

Dumpsgate

Question 13

Multiple choice · Domain: D2 Implementation and Integration › Implement GenAI application architectures

A financial services company wants to use Amazon Bedrock foundation models (FMs) to analyze call center recordings. When calls end, the call center stores recordings as MP3 files in an Amazon S3 bucket. The company needs to generate summaries and sentiment analysis for the recordings in a structured format as soon as new files are created. The recordings average 20 MB in size. Which combination of solutions will meet these requirements? (Select TWO.)

- A.** Use AWS Step Functions to orchestrate a workflow to process the recordings. Configure steps to invoke Amazon Transcribe to convert audio to text, validate job completion, and to invoke an AWS Lambda function to process the text by using Amazon Bedrock FMs to generate structured analysis output.
- B.** Use AWS Step Functions to orchestrate a workflow to process the recordings. Configure steps to invoke Amazon Transcribe to convert audio to text, validate job completion, and to directly invoke Amazon Bedrock FMs to generate summaries and sentiment analysis in JSON format.
- C.** Use AWS Step Functions to orchestrate a workflow to process the recordings. Configure steps to invoke Amazon Transcribe to convert audio to text, validate job completion, and to invoke an AWS Lambda function to create a prompt to invoke Amazon Bedrock FMs to generate structured analysis output.
- D.** Configure the source S3 bucket to send events to Amazon EventBridge. Create an EventBridge rule to invoke the Step Functions workflow when an object is created in the bucket.
- E.** Configure the source S3 bucket to send notifications to the Step Functions workflow when an object is created in the bucket.

Answer

- **B. Use AWS Step Functions to orchestrate a workflow to process the recordings. Configure steps to invoke Amazon Transcribe to convert audio to text, validate job completion, and to directly invoke Amazon Bedrock FMs to generate summaries and sentiment analysis in JSON format.**
- **D. Configure the source S3 bucket to send events to Amazon EventBridge. Create an EventBridge rule to invoke the Step Functions workflow when an object is created in the bucket.**

Explanation

Amazon Transcribe can convert each newly stored MP3 recording into text, which the workflow can then pass directly to an Amazon Bedrock model through Step Functions to produce structured summaries and sentiment analysis. Using Step Functions' Bedrock integration avoids adding a Lambda function just to construct or send the model request. Amazon S3 can publish object-created events to EventBridge, which can use a rule to start the workflow as soon as a recording is created.

<https://docs.aws.amazon.com/step-functions/latest/dg/connect-bedrock.html>

<https://docs.aws.amazon.com/AmazonS3/latest/userguide/EventBridge.html>

<https://docs.aws.amazon.com/step-functions/latest/dg/tutorial-cloudwatch-events-s3.html>

<https://docs.aws.amazon.com/step-functions/latest/dg/eventbridge-integration.html>

X A Use AWS Step Functions to orchestrate a workflow to process the recordings. Configure steps to invoke Amazon Transcribe to convert audio to text, validate job completion, and to invoke an AWS Lambda function to process the text by using Amazon Bedrock FMs to generate structured analysis output.

This adds a Lambda function to process the transcript and call Bedrock, although Step Functions can invoke Bedrock directly. The Lambda-based approach is not the intended, most direct combination for this requirement.

<https://docs.aws.amazon.com/step-functions/latest/dg/connect-bedrock.html>

<https://docs.aws.amazon.com/lambda/latest/dg/with-step-functions.html>

✓ **B** Use AWS Step Functions to orchestrate a workflow to process the recordings. Configure steps to invoke Amazon Transcribe to convert audio to text, validate job completion, and to directly invoke Amazon Bedrock FMs to generate summaries and sentiment analysis in JSON format. Transcribe converts the MP3 audio into text, and the workflow can wait for transcription to finish before passing the transcript to a Bedrock model. Step Functions supports direct Bedrock integration, so the model can generate the requested structured JSON without an extra Lambda function.

<https://docs.aws.amazon.com/step-functions/latest/dg/connect-bedrock.html>

<https://docs.aws.amazon.com/step-functions/latest/dg/sample-bedrock-prompt-chaining.html>

✗ **C** Use AWS Step Functions to orchestrate a workflow to process the recordings. Configure steps to invoke Amazon Transcribe to convert audio to text, validate job completion, and to invoke an AWS Lambda function to create a prompt to invoke Amazon Bedrock FMs to generate structured analysis output.

This inserts a Lambda function to create and send the Bedrock prompt, rather than using Step Functions' direct Bedrock integration. That extra component is unnecessary for the requested transcript analysis and structured output.

<https://docs.aws.amazon.com/step-functions/latest/dg/connect-bedrock.html>

<https://docs.aws.amazon.com/lambda/latest/dg/with-step-functions.html>

✓ **D** Configure the source S3 bucket to send events to Amazon EventBridge. Create an EventBridge rule to invoke the Step Functions workflow when an object is created in the bucket. S3 can send object-created events to EventBridge, and an EventBridge rule can start the Step Functions workflow when a new recording arrives. This provides the event-driven trigger needed to begin processing promptly.

<https://docs.aws.amazon.com/AmazonS3/latest/userguide/EventBridge.html>

<https://docs.aws.amazon.com/step-functions/latest/dg/tutorial-cloudwatch-events-s3.html>

<https://docs.aws.amazon.com/step-functions/latest/dg/eventbridge-integration.html>

✗ **E** Configure the source S3 bucket to send notifications to the Step Functions workflow when an object is created in the bucket.

S3 event notifications do not directly target a Step Functions workflow. Use EventBridge to receive the S3 object-created event and start the workflow instead.

<https://docs.aws.amazon.com/AmazonS3/latest/userguide/EventBridge.html>

<https://docs.aws.amazon.com/AmazonS3/latest/userguide/notification-how-to-event-types-and-destinations.html>

Question 14

Single choice · Domain: D2 Implementation and Integration › Implement deployment strategies for GenAI applications

A company is developing three specialized NLP models that support a customer service application. One model categorizes each customer's specific issue. Another model extracts key information from the customer interactions. The third model generates responses. The company must ensure that the application achieves at least 95% accuracy for all tasks. The application must handle up to 500 concurrent requests and respond in less than 500 ms during daily 2-hour peak usage periods. The company must ensure that the application optimizes resource usage during periods of low demand between usage spikes. Which solution will meet these requirements?

- A.** Deploy all three models to a single Amazon SageMaker AI multi-model endpoint. Enable dynamic scaling on the endpoint. Use a compute optimized instance type. Configure auto scaling policies that are based on invocation metrics to handle peak loads.
- B.** Deploy each model to a separate Amazon SageMaker Serverless Inference endpoint. Set provisioned concurrency to handle peak loads. Configure maximum concurrency limits and memory sizing based on each model's specific requirements.
- C.** Deploy the models by using Amazon Bedrock with provisioned throughput to handle peak loads. Configure the number of model units (MUs) based on expected token throughput needs. Implement request batching for each model.
- D.** Deploy each model to a separate Amazon SageMaker AI endpoint. Use an asynchronous inference configuration. Store model requests and responses in Amazon S3. Use Amazon SNS to send alerts to users when the application finishes processing requests.

Answer: B. Deploy each model to a separate Amazon SageMaker Serverless Inference endpoint. Set provisioned concurrency to handle peak loads. Configure maximum concurrency limits and memory sizing based on each model's specific requirements.

Explanation

Option B best matches the workload's bursty pattern: SageMaker Serverless Inference can scale capacity with demand, avoiding the need to keep endpoint instances running through low-demand periods. Provisioned concurrency can reserve capacity for the daily peak, while concurrency limits and model-specific memory sizing let the company tune each endpoint for its model and expected load. Separate endpoints also allow the classification, information-extraction, and response-generation models to be configured independently. The latency and accuracy targets still require validation and appropriate model sizing; the deployment choice alone does not guarantee them.

https://docs.aws.amazon.com/en_en/sagemaker/latest/dg/serverless-endpoints.html

<https://docs.aws.amazon.com/sagemaker/latest/dg/serverless-endpoints-monitoring.html>

<https://docs.aws.amazon.com/sagemaker/latest/dg/hosting-faqs.html>

✗ **A** Deploy all three models to a single Amazon SageMaker AI multi-model endpoint. Enable dynamic scaling on the endpoint. Use a compute optimized instance type. Configure auto scaling policies that are based on invocation metrics to handle peak loads.

A multi-model endpoint can host multiple models and use auto scaling, but it relies on provisioned endpoint instances rather than scaling to zero during low demand. That makes it a weaker fit for optimizing resource use between the daily spikes. The multi-model endpoint documentation describes this hosting approach and its instance requirements.

<https://docs.aws.amazon.com/sagemaker/latest/dg/multi-model-endpoints.html>

<https://docs.aws.amazon.com/sagemaker/latest/dg/multi-model-endpoint-instance.html>

✓ **B** Deploy each model to a separate Amazon SageMaker Serverless Inference endpoint. Set provisioned

concurrency to handle peak loads. Configure maximum concurrency limits and memory sizing based on each model's specific requirements.

Serverless Inference scales endpoint capacity with incoming traffic, which helps avoid paying for idle instance capacity between usage spikes. Provisioned concurrency can keep capacity ready for peak periods, while memory and maximum concurrency settings can be tailored to each model. These configurations support the stated burst and latency needs, though the company must test that its specific models meet the accuracy and sub-500-ms targets.

https://docs.aws.amazon.com/en_en/sagemaker/latest/dg/serverless-endpoints.html

<https://docs.aws.amazon.com/sagemaker/latest/dg/serverless-endpoints-monitoring.html>

<https://docs.aws.amazon.com/sagemaker/latest/dg/hosting-faqs.html>

✗ C Deploy the models by using Amazon Bedrock with provisioned throughput to handle peak loads. Configure the number of model units (MUs) based on expected token throughput needs. Implement request batching for each model.

Provisioned throughput and model units are intended to provide reserved model capacity, but this option does not address optimizing resource usage during low-demand intervals as directly as serverless endpoints. Request batching is also not established here as a way to meet the required sub-500-ms response time, since batching may add waiting time.

<https://docs.aws.amazon.com/bedrock/latest/userguide/prov-throughput.html>

https://docs.aws.amazon.com/us_en/bedrock/latest/userguide/prov-thru-purchase.html

https://docs.aws.amazon.com/bedrock/latest/APIReference/API_CustomModelUnits.html

✗ D Deploy each model to a separate Amazon SageMaker AI endpoint. Use an asynchronous inference configuration. Store model requests and responses in Amazon S3. Use Amazon SNS to send alerts to users when the application finishes processing requests.

Asynchronous inference is designed for requests that can be processed asynchronously, with outputs stored in S3 and notifications used to indicate completion. A customer-service application requiring responses in less than 500 ms needs synchronous, low-latency responses; waiting for an async job to finish does not fit that interaction pattern.

https://docs.aws.amazon.com/en_en/sagemaker/latest/dg/async-inference.html

<https://docs.aws.amazon.com/sagemaker/latest/dg/async-inference-check-predictions.html>

Question 15

Single choice · Domain: D2 Implementation and Integration › Integrate FMs with AWS services

A company is using Amazon Bedrock to develop an AI-powered application that uses a foundation model that supports cross-Region inference and provisioned throughput. The application must serve users in Europe and North America with consistently low latency. The application must comply with data residency regulations that require European user data to remain within Europe-based AWS Regions.

During testing, the application experiences service degradation when Regional traffic spikes reach service quotas. The company needs a solution that maintains application resilience and minimizes operational complexity.

Which solution will meet these requirements?

- A.** Deploy separate Amazon Bedrock instances in North American and European Regions. Use a custom routing layer that directs traffic based on user location. Configure Amazon CloudWatch alarms to monitor Regional service usage. Use Amazon SNS to send email alerts to the company when usage approaches specified thresholds.
- B.** Use Amazon Bedrock cross-Region inference profiles by specifying geographical codes in profile IDs when the application calls the InvokeModel API. Configure separate Amazon API Gateway HTTP APIs to direct European and North American users to the appropriate Regional endpoints.
- C.** Deploy a multi-Region Amazon API Gateway HTTP API and AWS Lambda functions that implement retry logic to handle throttling. Configure the Lambda functions to call the foundation model in the nearest secondary Region when the application reaches service quotas in the primary Region. Use intelligent routing to ensure compliance with data residency requirements.
- D.** Configure provisioned throughput for Amazon Bedrock in multiple Regions. Implement failover logic in the application code to switch between Regions when throttling occurs. Use AWS Global Accelerator to route traffic to the appropriate endpoints based on user location.

Answer: B. Use Amazon Bedrock cross-Region inference profiles by specifying geographical codes in profile IDs when the application calls the InvokeModel API. Configure separate Amazon API Gateway HTTP APIs to direct European and North American users to the appropriate Regional endpoints.

Explanation

Use geography-scoped Amazon Bedrock cross-Region inference profiles so requests can be distributed across supported Regions within the required geography, helping handle Regional traffic spikes while keeping European user data in Europe. Separate API Gateway HTTP APIs can direct European and North American users to the appropriate Regional endpoints. This uses Bedrock's managed inference routing rather than requiring custom failover logic, reducing operational complexity. [1][2][4]

<https://docs.aws.amazon.com/bedrock/latest/userguide/cross-region-inference.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/inference-profiles-support.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/inference-profiles-use.html>

✗ A Deploy separate Amazon Bedrock instances in North American and European Regions. Use a custom routing layer that directs traffic based on user location. Configure Amazon CloudWatch alarms to monitor Regional service usage. Use Amazon SNS to send email alerts to the company when usage approaches specified thresholds.

The custom routing layer and alerts can help operators notice Regional usage, but alerts do not automatically shift inference traffic when a Region reaches its quota. Building and maintaining separate

routing and failover behavior also adds operational complexity compared with managed cross-Region inference. [12]

<https://docs.aws.amazon.com/bedrock/latest/userguide/quotas-runtime.html>

✓ **B** Use Amazon Bedrock cross-Region inference profiles by specifying geographical codes in profile IDs when the application calls the InvokeModel API. Configure separate Amazon API Gateway HTTP APIs to direct European and North American users to the appropriate Regional endpoints.

Geography-scoped inference profiles allow Bedrock to route inference across supported Regions within the selected geography, helping absorb traffic spikes while respecting the European data-residency requirement. Regional API Gateway endpoints can direct users to the appropriate geography, and the managed profile avoids application-built cross-Region failover logic. [1][2][4]

<https://docs.aws.amazon.com/bedrock/latest/userguide/cross-region-inference.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/inference-profiles-support.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/inference-profiles-use.html>

✗ **C** Deploy a multi-Region Amazon API Gateway HTTP API and AWS Lambda functions that implement retry logic to handle throttling. Configure the Lambda functions to call the foundation model in the nearest secondary Region when the application reaches service quotas in the primary Region. Use intelligent routing to ensure compliance with data residency requirements.

A fallback to the nearest secondary Region is not necessarily constrained to Europe, so it does not by itself guarantee that European user data remains within Europe. Lambda-based retries and custom fallback routing also introduce operational complexity that managed geography-scoped inference profiles can avoid. [1][7]

<https://docs.aws.amazon.com/bedrock/latest/userguide/cross-region-inference.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/global-cross-region-inference.html>

✗ **D** Configure provisioned throughput for Amazon Bedrock in multiple Regions. Implement failover logic in the application code to switch between Regions when throttling occurs. Use AWS Global Accelerator to route traffic to the appropriate endpoints based on user location.

Provisioned throughput can provide reserved model capacity, but application-coded regional failover still adds operational complexity and does not inherently enforce Europe-only processing for European users. Global Accelerator's user-location routing alone does not provide Bedrock's geography-scoped inference routing. [8]

<https://docs.aws.amazon.com/bedrock/latest/userguide/prov-throughput.html>

Question 16

Single choice · Domain: D2 Implementation and Integration › Implement GenAI application architectures

An ecommerce company operates a global product recommendation system that needs to switch between multiple foundation models (FM) in Amazon Bedrock based on regulations, cost optimization, and performance requirements. The company must apply custom controls based on proprietary business logic, including dynamic cost thresholds, AWS Region-specific compliance rules, and real-time A/B testing across multiple FMs.

The system must be able to switch between FMs without deploying new code. The system must route user requests based on complex rules including user tier, transaction value, regulatory zone, and real-time cost metrics that change hourly and require immediate propagation across thousands of concurrent requests.

Which solution will meet these requirements?

- A.** Deploy an AWS Lambda function that uses environment variables to store routing rules and Amazon Bedrock FM IDs. Use the Lambda console to update the environment variables when business requirements change. Configure an Amazon API Gateway REST API to read request parameters to make routing decisions.
- B.** Deploy Amazon API Gateway REST API request transformation templates to implement routing logic based on request attributes. Store Amazon Bedrock FM endpoints as REST API stage variables. Update the variables when the system switches between models.
- C.** Configure an AWS Lambda function to fetch routing configurations from the AWS AppConfig Agent for each user request. Run business logic in the Lambda function to select the appropriate FM for each request. Expose the FM through a single Amazon API Gateway REST API endpoint.
- D.** Use AWS Lambda authorizers for an Amazon API Gateway REST API to evaluate routing rules that are stored in AWS AppConfig. Return authorization contexts based on business logic. Route requests to model-specific Lambda functions for each Amazon Bedrock FM.

Answer: C. Configure an AWS Lambda function to fetch routing configurations from the AWS AppConfig Agent for each user request. Run business logic in the Lambda function to select the appropriate FM for each request. Expose the FM through a single Amazon API Gateway REST API endpoint.

Explanation

C centralizes routing configuration in AWS AppConfig and lets the Lambda function apply proprietary rules to each request, including user tier, transaction value, Region compliance, and changing cost thresholds. The AppConfig Agent provides the function access to configuration without embedding routing rules in code, so rules and FM selections can change without deploying new code. A single API endpoint can use the Lambda routing logic to select among the FMs, including for A/B testing.

<https://docs.aws.amazon.com/appconfig/latest/userguide/appconfig-agent.html>

<https://docs.aws.amazon.com/appconfig/latest/userguide/appconfig-integration-lambda-extensions-how-it-works.html>

https://docs.aws.amazon.com/en_us/appconfig/latest/userguide/appconfig-agent-how-to-use.html

<https://docs.aws.amazon.com/appconfig/latest/userguide/>

X A Deploy an AWS Lambda function that uses environment variables to store routing rules and Amazon Bedrock FM IDs. Use the Lambda console to update the environment variables when business requirements change. Configure an Amazon API Gateway REST API to read request parameters to make routing decisions.

Lambda environment variables can hold configuration, but using them for frequently changing, complex routing rules makes updates cumbersome and couples configuration changes to Lambda function configuration management. API Gateway request parameters alone are also not a good place to implement the required business logic and changing cost-based decisions.

https://docs.aws.amazon.com/us_en/lambda/latest/dg/configuration-envvars.html

✗ **B** Deploy Amazon API Gateway REST API request transformation templates to implement routing logic based on request attributes. Store Amazon Bedrock FM endpoints as REST API stage variables. Update the variables when the system switches between models.

API Gateway mapping templates transform requests and stage variables can supply endpoint values, but this approach is a poor fit for complex business rules and hourly changing cost metrics. It also focuses on switching endpoint variables rather than evaluating all the request-specific conditions and selecting an FM in application logic.

<https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-mapping-variable-examples.html>

<https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-mapping-template-reference.html>

<https://docs.aws.amazon.com/apigateway/latest/developerguide/rest-api-data-transformations.html>

✓ **C** Configure an AWS Lambda function to fetch routing configurations from the AWS AppConfig Agent for each user request. Run business logic in the Lambda function to select the appropriate FM for each request. Expose the FM through a single Amazon API Gateway REST API endpoint.

AppConfig lets routing configuration be managed separately from Lambda code, while the AppConfig Agent provides the function access to that configuration. Lambda can evaluate the request attributes and current configuration to choose an FM, supporting custom routing without deploying new code when the rules or model selection change. A single API endpoint keeps clients independent of which FM is selected.

<https://docs.aws.amazon.com/appconfig/latest/userguide/appconfig-agent.html>

<https://docs.aws.amazon.com/appconfig/latest/userguide/appconfig-integration-lambda-extensions-how-it-works.html>

https://docs.aws.amazon.com/en_us/appconfig/latest/userguide/appconfig-agent-how-to-use.html

<https://docs.aws.amazon.com/appconfig/latest/userguide/>

✗ **D** Use AWS Lambda authorizers for an Amazon API Gateway REST API to evaluate routing rules that are stored in AWS AppConfig. Return authorization contexts based on business logic. Route requests to model-specific Lambda functions for each Amazon Bedrock FM.

A Lambda authorizer is intended to authorize API requests and provide authorization context, not to serve as the core per-request model-routing mechanism. Routing to separate model-specific functions also adds model-specific endpoints and functions, rather than selecting an FM through the single endpoint and application logic described in C.

<https://docs.aws.amazon.com/apigateway/latest/developerguide/http-api-lambda-authorizer.html>

Question 17

Single choice · Domain: D2 Implementation and Integration › Implement deployment strategies for GenAI applications

A company is implementing a serverless inference API by using AWS Lambda. The API will dynamically invoke multiple AI models hosted on Amazon Bedrock. The company needs to design a solution that can switch between model providers without modifying or redeploying Lambda code in real time. The design must include safe rollout of configuration changes and validation and rollback capabilities.

Which solution will meet these requirements?

- A.** Store the active model provider in AWS Systems Manager Parameter Store. Configure a Lambda function to read the parameter at runtime to determine which model to invoke.
- B.** Store the active model provider in AWS AppConfig. Configure a Lambda function to read the configuration at runtime to determine which model to invoke.
- C.** Configure an Amazon API Gateway REST API to route requests to separate Lambda functions. Hardcode each Lambda function to a specific model provider. Switch the integration target manually.
- D.** Store the active model provider in a JSON file hosted on Amazon S3. Use AWS AppConfig to reference the S3 file as a hosted configuration source. Configure a Lambda function to read the file through AppConfig at runtime to determine which model to invoke.

Answer: B. Store the active model provider in AWS AppConfig. Configure a Lambda function to read the configuration at runtime to determine which model to invoke.

Explanation

AWS AppConfig is designed to manage runtime configuration changes independently of Lambda code, and its Lambda integration lets the function retrieve the active configuration at runtime. AppConfig deployments can use validators and deployment strategies, including rollback capabilities, to make provider changes safer. This meets the requirements more directly than Parameter Store, a manually switched API Gateway integration, or the S3-backed setup as described.

<https://docs.aws.amazon.com/appconfig/latest/userguide/appconfig-creating-configuration-and-profile-validators.html>

<https://docs.aws.amazon.com/appconfig/latest/userguide/appconfig-deploying-reverting.html>

<https://docs.aws.amazon.com/appconfig/latest/userguide/appconfig-integration-lambda-extensions.html>

✗ **A** Store the active model provider in AWS Systems Manager Parameter Store. Configure a Lambda function to read the parameter at runtime to determine which model to invoke.

Parameter Store can hold a value that Lambda reads at runtime, so it can support changing the selected provider without redeploying code. However, this option does not provide the AppConfig deployment workflow for validating and safely rolling out configuration changes with rollback.

<https://docs.aws.amazon.com/systems-manager/latest/userguide/systems-manager-parameter-store.html>

<https://docs.aws.amazon.com/systems-manager/latest/userguide/ps-integration-lambda-extensions.html>

<https://docs.aws.amazon.com/systems-manager/latest/userguide/what-is-a-parameter.html>

✓ **B** Store the active model provider in AWS AppConfig. Configure a Lambda function to read the configuration at runtime to determine which model to invoke.

AppConfig lets the company change the provider selection independently of Lambda code, while the Lambda integration enables runtime retrieval of the configuration. AppConfig also supports configuration validators and controlled deployments with rollback, addressing the required safeguards.

<https://docs.aws.amazon.com/appconfig/latest/userguide/appconfig-creating-configuration-and-profile-validators.html>

<https://docs.aws.amazon.com/appconfig/latest/userguide/appconfig-deploying-reverting.html>

<https://docs.aws.amazon.com/appconfig/latest/userguide/appconfig-integration-lambda-extensions.html>

X C Configure an Amazon API Gateway REST API to route requests to separate Lambda functions. Hardcode each Lambda function to a specific model provider. Switch the integration target manually. Each function is tied to a provider in code, and changing the API Gateway integration target manually is not a runtime configuration change managed independently of those functions. This approach also does not provide AppConfig-style configuration validation and rollback for provider changes.

X D Store the active model provider in a JSON file hosted on Amazon S3. Use AWS AppConfig to reference the S3 file as a hosted configuration source. Configure a Lambda function to read the file through AppConfig at runtime to determine which model to invoke.

As written, this describes an S3 object as an AppConfig hosted configuration source; those are distinct configuration-source approaches. The direct AppConfig configuration in B is the match for managing the active provider setting through AppConfig's runtime integration and deployment safeguards, rather than adding an S3-backed configuration source.

<https://docs.aws.amazon.com/appconfig/latest/userguide/>

<https://docs.aws.amazon.com/appconfig/latest/userguide/appconfig-deploying-reverting.html>

<https://docs.aws.amazon.com/appconfig/latest/userguide/appconfig-integration-lambda-extensions.html>

Question 18

Single choice · Domain: D3 AI Safety, Security, and Governance › Implement responsible AI practices

A healthcare company is deploying an AI system that uses a foundation model (FM) to help clinicians make diagnostic decisions. The company's ethics board requires the AI system to demonstrate fairness across patient demographic groups and comply with medical AI governance policies. During initial testing, the AI system provides recommendations without clear explanations or decision tracing. Clinicians are unable to review how the AI system produces diagnostic conclusions.

The company needs to implement a solution that provides transparent reasoning for AI outputs, enables systematic fairness testing, and ensures policy compliance for responsible AI use in healthcare settings. The solution must balance comprehensive explainability with real-time performance requirements. The solution must support rapid iteration for bias testing across multiple demographic variables. The solution must integrate seamlessly with existing clinical workflows while maintaining strict data privacy controls. The solution must handle complex medical and regulatory terminology.

Which solution will meet these requirements?

- A.** Use Amazon SageMaker Clarify to generate model explanations. Use Amazon Augmented AI (Amazon A2I) to implement human review workflows. Use AWS Config to enforce compliance policies across the AI system.
- B.** Use Amazon Comprehend Medical to analyze medical terminology. Use Amazon Textract to process documents. Use AWS CloudFormation to standardize deployment configurations.
- C.** Use Amazon Bedrock agent tracing to provide reasoning traces. Use Amazon Bedrock Prompt Management with A/B testing to perform fairness evaluations. Use Amazon Bedrock Guardrails to ensure policy compliance.
- D.** Use Amazon CloudWatch to collect performance metrics. Use Amazon EventBridge to trigger compliance checks. Use AWS Lambda functions to generate custom explanation reports.

Answer: C. Use Amazon Bedrock agent tracing to provide reasoning traces. Use Amazon Bedrock Prompt Management with A/B testing to perform fairness evaluations. Use Amazon Bedrock Guardrails to ensure policy compliance.

Dumpsgate

Question 19

Multiple choice · Domain: D3 AI Safety, Security, and Governance › Implement security controls for GenAI applications

A financial services company is developing an AI-powered search assistant application to help investment advisors quickly retrieve investment data. The application runs as an AWS Lambda function. The company is using Amazon Bedrock to develop the application by using an Amazon Bedrock knowledge base that uses Amazon OpenSearch Serverless as its data source. The application agent must manage collections at scale by automatically assigning access permissions to collections and indexes that match a specific pattern. The company uses Amazon Bedrock tools to test the knowledge base. The knowledge base sync process finishes successfully. However, the test reveals a 400 Bad Authorization error from the BedrockAgentRuntime API and a 403 Forbidden error when the test attempts to access OpenSearch Serverless. The company must resolve the permissions issues.

Which combination of solutions will meet this requirement? (Select TWO.)

- A. Update the Lambda function execution role to include the bedrock:InvokeAgent permission. Add the aoss:APIAccessAll permission to the Lambda execution role.
- B. Create an OpenSearch Serverless data access policy that includes pattern-based resource rules.
- C. Configure a VPC endpoint policy for OpenSearch Serverless. Add the endpoint to the Lambda function's VPC configuration.
- D. Configure AWS Secrets Manager to store OpenSearch Serverless credentials. Grant the Lambda function access to retrieve the credentials.
- E. Enable IAM authentication for the OpenSearch Serverless domain. Add the es:ESHttp* permission to the Lambda function execution role.

Answer

- A. **Update the Lambda function execution role to include the bedrock:InvokeAgent permission. Add the aoss:APIAccessAll permission to the Lambda execution role.**
- B. **Create an OpenSearch Serverless data access policy that includes pattern-based resource rules.**

Explanation

The 400 authorization error requires the Lambda execution role to be authorized to invoke the Bedrock agent, while access to the OpenSearch Serverless data plane also requires the appropriate `aoss:APIAccessAll` IAM permission. OpenSearch Serverless additionally checks its data access policy, so create one with pattern-based rules that grant access to the required collections and indexes. Together, these address the Bedrock invocation and OpenSearch authorization failures while allowing permissions to scale across matching resources.

<https://docs.aws.amazon.com/opensearch-service/latest/developerguide/security-iam-serverless.html>
<https://docs.aws.amazon.com/bedrock/latest/userguide/kb-create-security.html>
https://docs.aws.amazon.com/opensearch-service/latest/developerguide/serverless-data-access.html?sc_campaign=datamlwave&sc_channel=el&sc_content=opensearch-kafka-golang&sc_country=mult&sc_geo=mult&sc_outcome=acq

✓ **A** Update the Lambda function execution role to include the bedrock:InvokeAgent permission. Add the aoss:APIAccessAll permission to the Lambda execution role.

The Lambda role needs permission to invoke the Bedrock agent, and `aoss:APIAccessAll` authorizes the role to make OpenSearch Serverless data-plane API calls. This IAM permission complements—not replaces—the OpenSearch Serverless data access policy in option B.

<https://docs.aws.amazon.com/opensearch-service/latest/developerguide/security-iam-serverless.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/kb-create-security.html>

https://docs.aws.amazon.com/service-authorization/latest/reference/list_opensearchserverless.html

✓ **B** Create an OpenSearch Serverless data access policy that includes pattern-based resource rules. OpenSearch Serverless uses data access policies to grant principals permissions on collections and indexes. Pattern-based resource rules let the agent manage access for collections and indexes that match the required naming pattern, addressing the 403 authorization failure at scale.

<https://docs.aws.amazon.com/cli/latest/reference/opensearchserverless/create-access-policy.html>

https://docs.aws.amazon.com/opensearch-service/latest/ServerlessAPIReference/API_CreateAccessPolicy.html

[https://docs.aws.amazon.com/opensearch-service/latest/developerguide/serverless-data-access.html?](https://docs.aws.amazon.com/opensearch-service/latest/developerguide/serverless-data-access.html?sc_campaign=datamlwave&sc_channel=el&sc_content=opensearch-kafka-golang&sc_country=mult&sc_geo=mult&sc_outcome=acq)

[sc_campaign=datamlwave&sc_channel=el&sc_content=opensearch-kafka-](https://docs.aws.amazon.com/opensearch-service/latest/developerguide/serverless-data-access.html?sc_campaign=datamlwave&sc_channel=el&sc_content=opensearch-kafka-golang&sc_country=mult&sc_geo=mult&sc_outcome=acq)

[golang&sc_country=mult&sc_geo=mult&sc_outcome=acq](https://docs.aws.amazon.com/opensearch-service/latest/developerguide/serverless-data-access.html?sc_campaign=datamlwave&sc_channel=el&sc_content=opensearch-kafka-golang&sc_country=mult&sc_geo=mult&sc_outcome=acq)

✗ **C** Configure a VPC endpoint policy for OpenSearch Serverless. Add the endpoint to the Lambda function's VPC configuration.

This changes network routing and endpoint-level controls, but the reported errors are authorization failures. The required fixes are IAM permissions for the caller and an OpenSearch Serverless data access policy, not adding a VPC endpoint.

✗ **D** Configure AWS Secrets Manager to store OpenSearch Serverless credentials. Grant the Lambda function access to retrieve the credentials.

OpenSearch Serverless access is controlled through IAM and data access policies; this option proposes storing credentials rather than granting the needed policy permissions. Retrieving credentials would not resolve the missing authorization rules described in the scenario.

<https://docs.aws.amazon.com/opensearch-service/latest/developerguide/security-iam-serverless.html>

[https://docs.aws.amazon.com/opensearch-service/latest/developerguide/serverless-data-access.html?](https://docs.aws.amazon.com/opensearch-service/latest/developerguide/serverless-data-access.html?sc_campaign=datamlwave&sc_channel=el&sc_content=opensearch-kafka-golang&sc_country=mult&sc_geo=mult&sc_outcome=acq)

[sc_campaign=datamlwave&sc_channel=el&sc_content=opensearch-kafka-](https://docs.aws.amazon.com/opensearch-service/latest/developerguide/serverless-data-access.html?sc_campaign=datamlwave&sc_channel=el&sc_content=opensearch-kafka-golang&sc_country=mult&sc_geo=mult&sc_outcome=acq)

[golang&sc_country=mult&sc_geo=mult&sc_outcome=acq](https://docs.aws.amazon.com/opensearch-service/latest/developerguide/serverless-data-access.html?sc_campaign=datamlwave&sc_channel=el&sc_content=opensearch-kafka-golang&sc_country=mult&sc_geo=mult&sc_outcome=acq)

✗ **E** Enable IAM authentication for the OpenSearch Serverless domain. Add the es:ESHttp* permission to the Lambda function execution role.

This describes permissions associated with OpenSearch Service domains, not the OpenSearch Serverless collection in this scenario. OpenSearch Serverless uses `aoss` permissions and data access policies, rather than `es:ESHttp\*` domain permissions.

<https://docs.aws.amazon.com/opensearch-service/latest/developerguide/security-iam-serverless.html>

https://docs.aws.amazon.com/service-authorization/latest/reference/list_opensearchserverless.html

Question 20

Single choice · Domain: D3 AI Safety, Security, and Governance › Implement content moderation and guardrails

A company is using Amazon Bedrock to develop a customer support AI assistant. The AI assistant must respond to customer questions about their accounts. The AI assistant must not expose personal information in responses. The company must comply with data residency policies by ensuring that all processing occurs within the same AWS Region where each customer is located.

The company wants to evaluate how effective the AI assistant is at preventing the exposure of personal information before the company makes the AI assistant available to customers.

Which solution will meet these requirements?

- A.** Configure a cross-Region Amazon Bedrock guardrail to apply sensitive information filters. Set the guardrail to detect mode during development and testing. Switch to block mode for production deployment.
- B.** Configure an Amazon Bedrock guardrail to apply sensitive information filters. Set the guardrail to mask mode during development and testing. Switch to block mode for production deployment. Deploy a copy of the guardrail to each Region where the company operates.
- C.** Configure an Amazon Bedrock guardrail to apply content and topic filters. Set the guardrail to detect mode during development, testing, and production. Disable invocation logging for the Amazon Bedrock model.
- D.** Configure a cross-Region Amazon Bedrock guardrail to apply a set of content and word filters. Set the guardrail to detect mode during development and testing. Switch to mask mode for production deployment.

Answer: B. Configure an Amazon Bedrock guardrail to apply sensitive information filters. Set the guardrail to mask mode during development and testing. Switch to block mode for production deployment. Deploy a copy of the guardrail to each Region where the company operates.

Explanation

Sensitive information filters target personal information, and mask mode lets the company assess the assistant's behavior during development and testing while obscuring detected information. Switching to block mode prevents responses containing detected sensitive information from being returned in production. Deploying a guardrail copy in each customer's Region avoids using a cross-Region configuration that could conflict with the data-residency requirement.

<https://docs.aws.amazon.com/bedrock/latest/userguide/guardrails-sensitive-filters.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/guardrails-harmful-content-handling-options.html>

https://docs.aws.amazon.com/us_en/bedrock/latest/userguide/guardrails-cross-region.html

<https://docs.aws.amazon.com/bedrock/latest/userguide/geographic-cross-region-inference.html>

✗ A Configure a cross-Region Amazon Bedrock guardrail to apply sensitive information filters. Set the guardrail to detect mode during development and testing. Switch to block mode for production deployment.

Sensitive information filters and the production block mode are relevant, but a cross-Region guardrail conflicts with the requirement that processing remain in the customer's Region. Detect mode reports detected information without preventing it from appearing in responses during testing; mask mode is the suitable choice when evaluating behavior while obscuring sensitive information.

<https://docs.aws.amazon.com/bedrock/latest/userguide/guardrails-sensitive-filters.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/guardrails-harmful-content-handling-options.html>

https://docs.aws.amazon.com/us_en/bedrock/latest/userguide/guardrails-cross-region.html

<https://docs.aws.amazon.com/bedrock/latest/userguide/geographic-cross-region-inference.html>

✓ **B** Configure an Amazon Bedrock guardrail to apply sensitive information filters. Set the guardrail to mask mode during development and testing. Switch to block mode for production deployment. Deploy a copy of the guardrail to each Region where the company operates.

Sensitive information filters address the risk of exposing personal information. Mask mode supports evaluation by masking detected sensitive information, and block mode can prevent responses containing detected information from reaching customers. Deploying a guardrail copy in each Region supports the company's requirement to keep processing in the customer's Region rather than using cross-Region processing.

<https://docs.aws.amazon.com/bedrock/latest/userguide/guardrails-sensitive-filters.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/guardrails-harmful-content-handling-options.html>

https://docs.aws.amazon.com/us_en/bedrock/latest/userguide/guardrails-cross-region.html

<https://docs.aws.amazon.com/bedrock/latest/userguide/geographic-cross-region-inference.html>

✗ **C** Configure an Amazon Bedrock guardrail to apply content and topic filters. Set the guardrail to detect mode during development, testing, and production. Disable invocation logging for the Amazon Bedrock model.

Content and topic filters do not specifically address the exposure of personal information; sensitive information filters are needed for that requirement. Detect mode does not block or mask detected information in production, and disabling invocation logging does not itself prevent personal information from appearing in responses.

<https://docs.aws.amazon.com/bedrock/latest/userguide/guardrails-sensitive-filters.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/guardrails-harmful-content-handling-options.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/guardrails-components.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/guardrails-content-filters-overview.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/data-retention.html>

✗ **D** Configure a cross-Region Amazon Bedrock guardrail to apply a set of content and word filters. Set the guardrail to detect mode during development and testing. Switch to mask mode for production deployment.

Content and word filters are not the specific sensitive information filters needed to address personal information in account-related responses. Mask mode obscures detected information rather than blocking the response, and using a cross-Region guardrail conflicts with the requirement to keep processing within each customer's Region.

<https://docs.aws.amazon.com/bedrock/latest/userguide/guardrails-sensitive-filters.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/guardrails-harmful-content-handling-options.html>

https://docs.aws.amazon.com/us_en/bedrock/latest/userguide/guardrails-cross-region.html

<https://docs.aws.amazon.com/bedrock/latest/userguide/geographic-cross-region-inference.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/guardrails-components.html>

Question 21

Single choice · Domain: D4 Operational Efficiency and Optimization › Implement scalability and reliability for GenAI applications

A media company is building an AI-powered content moderation system by using Amazon Bedrock. The system first classifies text by using a small, low-latency model. Then the system escalates requests that have a confidence score below 0.65 to a larger, more expensive model.

The system must respond in near real time for high-confidence results. The system must process lowconfidence requests asynchronously. The system must scale to meet sudden spikes in demand. The company wants to optimize costs for the system by invoking the larger model only when required.

The company wants to use decoupled components to achieve high resiliency for the system.

Which solution will meet these requirements?

- A.** Use Amazon API Gateway to invoke the small model synchronously. If the small model's confidence score is below 0.65, synchronously call the larger model. Use provisioned concurrency to handle traffic spikes.
- B.** Use an AWS Step Functions workflow that has parallel branches to run both the small model and the large model for every request. Choose the large model result when confidence score values differ.
- C.** Send requests to an Amazon SQS queue. Use AWS Fargate to process messages. Invoke the small model first. If the confidence score is below 0.65, place the request in a second SQS queue to process asynchronously by using the large model.
- D.** Deploy both models on Amazon EC2 instances and enable auto scaling. Use a custom application heuristic to route requests to the appropriate instance based on phrase length and keyword rules.

Answer: C. Send requests to an Amazon SQS queue. Use AWS Fargate to process messages. Invoke the small model first. If the confidence score is below 0.65, place the request in a second SQS queue to process asynchronously by using the large model.

Explanation

Option C uses queues to decouple request intake and processing, helping the system absorb sudden demand spikes and remain resilient. A worker can invoke the small model first and send only requests below the 0.65 confidence threshold to a second queue for asynchronous processing by the larger model. This preserves near-real-time handling for high-confidence requests while avoiding unnecessary invocations of the more expensive model. Amazon SQS supports queued message processing, and ECS services can scale based on queue demand.

<https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/welcome.html>

https://docs.aws.amazon.com/sqs/?icmpid=docs_homepage_serverless

<https://docs.aws.amazon.com/AmazonECS/latest/developerguide/service-autoscaling-queue.html>

X A Use Amazon API Gateway to invoke the small model synchronously. If the small model's confidence score is below 0.65, synchronously call the larger model. Use provisioned concurrency to handle traffic spikes.

This routes low-confidence requests to the larger model synchronously, so those requests are not processed asynchronously as required. Synchronous calls also couple request handling to model response times rather than buffering demand with a queue.

https://docs.aws.amazon.com/us_en/bedrock/latest/userguide/inference-api.html

<https://docs.aws.amazon.com/bedrock/latest/userguide/apis.html>

✗ **B** Use an AWS Step Functions workflow that has parallel branches to run both the small model and the large model for every request. Choose the large model result when confidence score values differ. The parallel branches invoke the larger model for every request, including high-confidence ones, which conflicts with the requirement to use it only when needed to control costs. Running both models concurrently also does not implement the required small-model-first escalation based on the 0.65 threshold.

<https://docs.aws.amazon.com/pdfs/step-functions/latest/dg/step-functions-dg.pdf>

https://docs.aws.amazon.com/hi_in/step-functions/latest/dg/state-parallel.html

✓ **C** Send requests to an Amazon SQS queue. Use AWS Fargate to process messages. Invoke the small model first. If the confidence score is below 0.65, place the request in a second SQS queue to process asynchronously by using the large model.

The first queue decouples incoming requests from processing, while workers can invoke the small model and send only low-confidence requests to the second queue. This supports asynchronous escalation and avoids invoking the larger, more expensive model for high-confidence results. Queue-based processing can also absorb spikes, and a container service can scale workers in response to queue demand.

<https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/welcome.html>

https://docs.aws.amazon.com/sqs/?icmpid=docs_homepage_serverless

<https://docs.aws.amazon.com/AmazonECS/latest/developerguide/service-autoscaling-queue.html>

✗ **D** Deploy both models on Amazon EC2 instances and enable auto scaling. Use a custom application heuristic to route requests to the appropriate instance based on phrase length and keyword rules. Phrase length and keyword rules do not route requests according to the small model's actual confidence score and the specified 0.65 threshold. The option also provides no queue-based asynchronous path for low-confidence requests, and deploying models on EC2 does not use Amazon Bedrock as specified.

https://docs.aws.amazon.com/us_en/bedrock/latest/userguide/inference-api.html

<https://docs.aws.amazon.com/bedrock/latest/userguide/apis.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/inference.html>

Question 22

Single choice · Domain: D4 Operational Efficiency and Optimization › Optimize performance of GenAI applications

A global financial services company hosts a fraud-alert system that uses an Amazon Bedrock foundation model (FM) to generate explanations for suspicious transactions. The company processes regulated financial data across three geographic areas. The system must maintain consistent responsiveness globally, support multi-Region failover, and provide full observability for audit and compliance teams.

Load testing shows that the FM's total inference time cannot be reduced. The company cannot increase its inference costs, change the FM, modify token counts, or provision additional compute capacity. Users report that the UI performs slowly because it waits for the complete model response before it shows any content.

The company must improve perceived responsiveness during peak periods, when the system can receive 10,000–15,000 concurrent requests. The solution must maintain multi-Region resiliency and full monitoring visibility.

Which solution will meet these requirements?

- A.** Enable response streaming by using the `InvokeModelWithResponseStream` API so the frontend can display generated tokens as the tokens arrive. Collect metrics in Amazon CloudWatch and enable distributed tracing to monitor streaming latency and Regional performance.
- B.** Deploy Regional Amazon Bedrock inference endpoints. Set up latency-based Amazon Route 53 routing. Cache partially processed explanations in a global Amazon DynamoDB table to serve responses more quickly during peak periods.
- C.** Use a Lambda@Edge preprocessing layer to condense inputs during peak periods. Asynchronously call Amazon Bedrock while the system returns interim placeholder responses to customers.
- D.** Deploy AWS Lambda functions to handle inference requests across multiple AWS Regions. Increase Lambda concurrency limits. Scale down Amazon CloudWatch Logs retention to reduce backend load during peak periods.

Answer: A. Enable response streaming by using the `InvokeModelWithResponseStream` API so the frontend can display generated tokens as the tokens arrive. Collect metrics in Amazon CloudWatch and enable distributed tracing to monitor streaming latency and Regional performance.

Explanation

The UI is slow because it waits for the complete inference result, even though the model's total inference time cannot be reduced. Response streaming lets the frontend show generated tokens as they arrive, improving perceived responsiveness without changing the model, token counts, or inference capacity. CloudWatch metrics and distributed tracing provide visibility into streaming latency and regional performance while leaving the existing multi-Region design intact.

<https://docs.aws.amazon.com/bedrock/latest/userguide/monitoring-mantle-metrics.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/trace-view.html>

https://docs.aws.amazon.com/bedrock/latest/APIReference/API_runtime_InvokeModelWithResponseStream.html

✓ **A** Enable response streaming by using the `InvokeModelWithResponseStream` API so the frontend can display generated tokens as the tokens arrive. Collect metrics in Amazon CloudWatch and enable distributed tracing to monitor streaming latency and Regional performance.

Streaming sends output incrementally, so users can see the explanation before the full response is ready;

it improves perceived responsiveness without reducing inference time or requiring more compute. CloudWatch metrics and tracing provide monitoring visibility, and this approach does not require replacing the existing multi-Region architecture.

<https://docs.aws.amazon.com/bedrock/latest/userguide/monitoring-mantle-metrics.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/trace-view.html>

https://docs.aws.amazon.com/bedrock/latest/APIReference/API_runtime_InvokeModelWithResponseStream.html

✗ B Deploy Regional Amazon Bedrock inference endpoints. Set up latency-based Amazon Route 53 routing. Cache partially processed explanations in a global Amazon DynamoDB table to serve responses more quickly during peak periods.

Latency-based routing can direct requests toward a lower-latency Region, but it does not make the UI display output before the full model response arrives. Caching partially processed explanations also does not address the stated cause of the delay and introduces a new data-handling mechanism for regulated transaction explanations.

<https://docs.aws.amazon.com/bedrock/latest/userguide/global-cross-region-inference.html>

<https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/routing-policy-latency-phz.html>

✗ C Use a Lambda@Edge preprocessing layer to condense inputs during peak periods. Asynchronously call Amazon Bedrock while the system returns interim placeholder responses to customers. Condensing inputs changes the model input and potentially the token count, which the company cannot do. Returning placeholders does not stream the actual explanation to users and does not provide the requested monitoring or multi-Region failover solution.

✗ D Deploy AWS Lambda functions to handle inference requests across multiple AWS Regions. Increase Lambda concurrency limits. Scale down Amazon CloudWatch Logs retention to reduce backend load during peak periods.

Increasing Lambda concurrency changes capacity provisioning, which is prohibited, and reducing log retention undermines audit and compliance needs. This option also does not change the UI's behavior of waiting for the complete model response.

Question 23

Single choice · Domain: D4 Operational Efficiency and Optimization › Implement monitoring and observability for GenAI applications

A financial services company processes more than 10,000 customer inquiries every day through a multi-agent GenAI application that uses Amazon Bedrock AgentCore. The application agents invoke several custom tools. During peak usage periods, users report that the custom tools experience up to 40% failure rates. The tools perform inconsistently for different teams at the company.

A GenAI developer must implement an observability solution that provides end-to-end visibility into agent interactions and tool behavior. The solution must use built-in Amazon Bedrock capabilities and must not require custom instrumentation. The GenAI developer needs a solution that requires minimal performance overhead.

Which solution will meet these requirements?

- A.** Enable AgentCore Observability and trace collection. Use AWS X-Ray to capture distributed traces for the custom tools. Build Amazon CloudWatch dashboards to visualize metrics for errors, throttling, and latency during peak usage periods.
- B.** Use Amazon CloudWatch Container Insights to monitor the agents. Configure an AWS Lambda function to poll the Amazon Bedrock API for tool usage metrics. Configure the function to store results in CloudWatch to generate alerts.
- C.** Build a custom ETL pipeline that uses AWS Lambda functions to process Amazon CloudWatch logs from Amazon Bedrock. Store the processed data in Amazon DynamoDB. Use Amazon QuickSight to visualize cross-team performance patterns.
- D.** Enable AgentCore Observability and send trace data to Amazon CloudWatch Logs. Use a custom AWS Lambda function to extract tool performance metrics from the logs. Use Amazon Managed Grafana to visualize trends.

Answer: A. Enable AgentCore Observability and trace collection. Use AWS X-Ray to capture distributed traces for the custom tools. Build Amazon CloudWatch dashboards to visualize metrics for errors, throttling, and latency during peak usage periods.

Explanation

AgentCore Observability provides built-in tracing for agent interactions and tool behavior, giving the developer end-to-end visibility without requiring a separate custom telemetry pipeline. Using AWS X-Ray for distributed traces and CloudWatch dashboards for error, throttling, and latency metrics helps identify the tools and teams affected during peak periods. This approach fits the requirements for built-in observability and low instrumentation overhead.

<https://docs.aws.amazon.com/bedrock-agentcore/latest/devguide/observability-view.html>

<https://docs.aws.amazon.com/bedrock-agentcore/latest/devguide/observability-configure.html>

<https://docs.aws.amazon.com/bedrock-agentcore/latest/devguide/observability-telemetry.html>

<https://docs.aws.amazon.com/xray/latest/devguide/xray-services-agentcore.html>

<https://docs.aws.amazon.com/bedrock-agentcore/latest/devguide/observability-runtime-metrics.html>

✓ **A** Enable AgentCore Observability and trace collection. Use AWS X-Ray to capture distributed traces for the custom tools. Build Amazon CloudWatch dashboards to visualize metrics for errors, throttling, and latency during peak usage periods.

AgentCore Observability and trace collection provide built-in visibility into agent interactions, while X-Ray can show distributed trace behavior across the application. CloudWatch dashboards make the relevant error, throttling, and latency patterns visible during peak usage, without introducing a custom polling or ETL pipeline.

<https://docs.aws.amazon.com/bedrock-agentcore/latest/devguide/observability-view.html>
<https://docs.aws.amazon.com/bedrock-agentcore/latest/devguide/observability-configure.html>
<https://docs.aws.amazon.com/bedrock-agentcore/latest/devguide/observability-telemetry.html>
<https://docs.aws.amazon.com/xray/latest/devguide/xray-services-agentcore.html>
<https://docs.aws.amazon.com/bedrock-agentcore/latest/devguide/observability-runtime-metrics.html>

✗ **B** Use Amazon CloudWatch Container Insights to monitor the agents. Configure an AWS Lambda function to poll the Amazon Bedrock API for tool usage metrics. Configure the function to store results in CloudWatch to generate alerts.

Container Insights focuses on container and infrastructure monitoring rather than end-to-end agent and custom-tool interactions. Polling with a Lambda function adds custom processing and recurring overhead, and does not provide the built-in tracing solution requested.

<https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/ContainerInsights.html>
<https://docs.aws.amazon.com/bedrock-agentcore/latest/devguide/observability-view.html>
<https://docs.aws.amazon.com/bedrock-agentcore/latest/devguide/observability-runtime-metrics.html>

✗ **C** Build a custom ETL pipeline that uses AWS Lambda functions to process Amazon CloudWatch logs from Amazon Bedrock. Store the processed data in Amazon DynamoDB. Use Amazon QuickSight to visualize cross-team performance patterns.

This option depends on custom Lambda-based log processing and a separate data pipeline rather than built-in AgentCore observability. It adds operational and performance overhead and does not directly provide the requested end-to-end traces of agent and tool interactions.

<https://docs.aws.amazon.com/bedrock-agentcore/latest/devguide/observability-view.html>
<https://docs.aws.amazon.com/bedrock-agentcore/latest/devguide/observability-configure.html>
<https://docs.aws.amazon.com/bedrock-agentcore/latest/devguide/observability-telemetry.html>

✗ **D** Enable AgentCore Observability and send trace data to Amazon CloudWatch Logs. Use a custom AWS Lambda function to extract tool performance metrics from the logs. Use Amazon Managed Grafana to visualize trends.

Although it enables AgentCore Observability, the custom Lambda function for extracting tool metrics adds custom processing that the requirements rule out and introduces additional overhead. The built-in tracing and CloudWatch metrics approach in option A avoids that extra extraction step.

<https://docs.aws.amazon.com/bedrock-agentcore/latest/devguide/observability-view.html>
<https://docs.aws.amazon.com/bedrock-agentcore/latest/devguide/observability-configure.html>
<https://docs.aws.amazon.com/bedrock-agentcore/latest/devguide/observability-telemetry.html>
<https://docs.aws.amazon.com/bedrock-agentcore/latest/devguide/observability-runtime-metrics.html>

Question 24

Single choice · Domain: D4 Operational Efficiency and Optimization › Implement monitoring and observability for GenAI applications

A company has a recommendation system. The system's applications run on Amazon EC2 instances. The applications make API calls to Amazon Bedrock foundation models (FMs) to analyze customer behavior and generate personalized product recommendations.

The system is experiencing intermittent issues. Some recommendations do not match customer preferences. The company needs an observability solution to monitor operational metrics and detect patterns of operational performance degradation compared to established baselines. The solution must also generate alerts with correlation data within 10 minutes when FM behavior deviates from expected patterns.

Which solution will meet these requirements?

- A.** Configure Amazon CloudWatch Container Insights for the application infrastructure. Set up CloudWatch alarms for latency thresholds. Add custom metrics for token counts by using the CloudWatch embedded metric format. Create CloudWatch dashboards to visualize the data.
- B.** Implement AWS X-Ray to trace requests through the application components. Enable CloudWatch Logs Insights for error pattern detection. Set up AWS CloudTrail to monitor all API calls to Amazon Bedrock. Create custom dashboards in Amazon QuickSight.
- C.** Enable Amazon CloudWatch Application Insights for the application resources. Create custom metrics for recommendation quality, token usage, and response latency by using the CloudWatch embedded metric format with dimensions for request types and user segments. Configure CloudWatch anomaly detection on the model metrics. Establish log pattern analysis by using CloudWatch Logs Insights.
- D.** Use Amazon OpenSearch Service with the Observability plugin. Ingest model metrics and logs by using Amazon Kinesis. Create custom Piped Processing Language (PPL) queries to analyze model behavior patterns. Establish operational dashboards to visualize anomalies in real time.

Answer: C. Enable Amazon CloudWatch Application Insights for the application resources. Create custom metrics for recommendation quality, token usage, and response latency by using the CloudWatch embedded metric format with dimensions for request types and user segments. Configure CloudWatch anomaly detection on the model metrics. Establish log pattern analysis by using CloudWatch Logs Insights.

Explanation

Option C collects metrics relevant to FM behavior, including recommendation quality, token usage, and response latency, and uses dimensions to distinguish request types and user segments. CloudWatch anomaly detection can identify deviations from learned baselines, while Application Insights helps detect and correlate operational issues across application resources. CloudWatch Logs Insights adds log pattern analysis, supporting investigation and alerts when behavior departs from expectations. [2, 5, 6, 11]

https://docs.aws.amazon.com/en_en/AmazonCloudWatch/latest/monitoring/CloudWatch_Embedded_Metric_Format_Specification.html

<https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/appinsights-how-works.html>

<https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/appinsights-what-is.html>

https://docs.aws.amazon.com/AmazonCloudWatch/latest/APIReference/API_AnomalyDetector.html

X A Configure Amazon CloudWatch Container Insights for the application infrastructure. Set up CloudWatch alarms for latency thresholds. Add custom metrics for token counts by using the CloudWatch

embedded metric format. Create CloudWatch dashboards to visualize the data.

Container Insights and latency-threshold alarms can help monitor infrastructure and operational latency, and the embedded metric format can publish token-count metrics. However, this option does not configure anomaly detection against established baselines or measure recommendation quality, so it does not adequately detect deviations in FM behavior.

https://docs.aws.amazon.com/en_en/AmazonCloudWatch/latest/monitoring/CloudWatch_Embedded_Metric_Format_Specification.html

✗ B Implement AWS X-Ray to trace requests through the application components. Enable CloudWatch Logs Insights for error pattern detection. Set up AWS CloudTrail to monitor all API calls to Amazon Bedrock. Create custom dashboards in Amazon QuickSight.

Tracing and log analysis can help investigate application requests and errors, while CloudTrail records API activity. But this solution does not define FM-behavior metrics or configure baseline-based anomaly detection and correlated alerts, which are central to the requirements.

https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/CWL_AnalyzeLogData_Patterns.html

<https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/LogAnalytics.html>

✓ C Enable Amazon CloudWatch Application Insights for the application resources. Create custom metrics for recommendation quality, token usage, and response latency by using the CloudWatch embedded metric format with dimensions for request types and user segments. Configure CloudWatch anomaly detection on the model metrics. Establish log pattern analysis by using CloudWatch Logs Insights. These custom metrics cover both recommendation outcomes and operational signals, with dimensions that help distinguish behavior across request types and user segments. CloudWatch anomaly detection compares metrics with expected patterns, and Application Insights and Logs Insights support detecting and investigating correlated operational issues. This combination best addresses the need to monitor degradation and alert on deviations from expected FM behavior.

https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/CWL_AnalyzeLogData_Patterns.html

https://docs.aws.amazon.com/en_en/AmazonCloudWatch/latest/monitoring/CloudWatch_Embedded_Metric_Format_Specification.html

<https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/appinsights-how-works.html>

<https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/appinsights-what-is.html>

https://docs.aws.amazon.com/AmazonCloudWatch/latest/APIReference/API_AnomalyDetector.html

✗ D Use Amazon OpenSearch Service with the Observability plugin. Ingest model metrics and logs by using Amazon Kinesis. Create custom Piped Processing Language (PPL) queries to analyze model behavior patterns. Establish operational dashboards to visualize anomalies in real time.

OpenSearch dashboards and queries could help visualize and explore ingested metrics and logs. The option does not specify baseline-based anomaly detection or alerting with correlated operational data within the required time, so it does not establish the requested detection-and-alerting solution.

Question 25

Single choice · Domain: D5 Testing, Validation, and Troubleshooting › Test GenAI applications

A healthcare company wants to develop a proof-of-concept application that uses Amazon Bedrock to automatically summarize medical documents. The company has 3 weeks to validate the application's accuracy. The application must comply with the company's data privacy policies. The application must include metrics to evaluate summarization accuracy and processing time. Which solution will meet these requirements?

- A.** Create a dataset that includes 50-100 anonymized patient records. Implement Retrieval Augmented Generation (RAG) with a secure knowledge base. Use a judge model to evaluate accuracy metrics across three foundation models (FMs).
- B.** Fine-tune a single foundation model (FM) on patient records. Deploy the FM on Amazon Bedrock. Use Amazon Bedrock AgentCore to configure the FM as an agent. Conduct user testing on 500 company staff members.
- C.** Select the most powerful available AWS foundation model (FM). Create a chat interface by using Converse APIs. Test the application on 50-100 actual patient records by using only qualitative feedback from stakeholders. Use a custom web interface to gather real-world performance metrics.
- D.** Use the Strands SDK to deploy multiple agents that connect to multiple knowledge bases that contain specialized medical documents. Compare the responses of the agents. Evaluate the integration of the agents with the company's existing systems.

Answer: A. Create a dataset that includes 50-100 anonymized patient records. Implement Retrieval Augmented Generation (RAG) with a secure knowledge base. Use a judge model to evaluate accuracy metrics across three foundation models (FMs).

Explanation

Option A is the best fit for a three-week proof of concept: a modest set of anonymized records can be used to test the summarization approach while a secure knowledge base supports retrieval from relevant documents. A judge model can provide a structured way to evaluate accuracy and compare three foundation models rather than relying only on subjective feedback. The team can also record processing time for each test as a performance metric. Amazon Bedrock documentation describes evaluation metrics and evaluation jobs, as well as knowledge bases and their security controls. [1][2][3][4][11][12]

<https://docs.aws.amazon.com/bedrock/latest/userguide/model-evaluation-metrics.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/model-evaluation-built-in-metrics.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/knowledge-base.html>

https://docs.aws.amazon.com/en_en/bedrock/latest/userguide/encryption-kb.html

https://docs.aws.amazon.com/us_en/bedrock/latest/userguide/model-evaluation-jobs-management-create.html

<https://docs.aws.amazon.com/bedrock/latest/userguide/evaluation.html>

✓ **A** Create a dataset that includes 50-100 anonymized patient records. Implement Retrieval Augmented Generation (RAG) with a secure knowledge base. Use a judge model to evaluate accuracy metrics across three foundation models (FMs).

Anonymizing the records and using a secure knowledge base address the company's privacy requirements, while the small dataset is practical for a short proof of concept. A judge model supports structured evaluation and comparison of accuracy across multiple FMs; processing time can be recorded during those same tests. Bedrock supports model evaluation and secure knowledge-base configurations. [1][2][3][4][11][12]

<https://docs.aws.amazon.com/bedrock/latest/userguide/model-evaluation-metrics.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/model-evaluation-built-in-metrics.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/knowledge-base.html>

https://docs.aws.amazon.com/en_en/bedrock/latest/userguide/encryption-kb.html

https://docs.aws.amazon.com/us_en/bedrock/latest/userguide/model-evaluation-jobs-management-create.html

<https://docs.aws.amazon.com/bedrock/latest/userguide/evaluation.html>

X B Fine-tune a single foundation model (FM) on patient records. Deploy the FM on Amazon Bedrock. Use Amazon Bedrock AgentCore to configure the FM as an agent. Conduct user testing on 500 company staff members.

The option uses patient records without saying they are anonymized or otherwise protected, so it does not clearly meet the privacy requirement. Fine-tuning and testing with 500 staff members also add work without a stated method for measuring summarization accuracy or processing time, and it evaluates only one FM. AgentCore can support agent runtime capabilities, but configuring an agent does not itself provide the required evaluation. [6]

<https://docs.aws.amazon.com/bedrock-agentcore/latest/devguide/agents-tools-runtime.html>

X C Select the most powerful available AWS foundation model (FM). Create a chat interface by using Converse APIs. Test the application on 50-100 actual patient records by using only qualitative feedback from stakeholders. Use a custom web interface to gather real-world performance metrics.

Testing on actual patient records does not establish that the company's privacy policies are being followed. Qualitative stakeholder feedback alone does not provide the required accuracy metrics, and choosing the "most powerful" model does not validate its suitability for this task. A custom interface may capture performance data, but it does not resolve the privacy concern or replace a defined accuracy evaluation. [7][9]

<https://docs.aws.amazon.com/bedrock/latest/userguide/apis.html>

https://docs.aws.amazon.com/en_us/bedrock/latest/userguide/models.html

X D Use the Strands SDK to deploy multiple agents that connect to multiple knowledge bases that contain specialized medical documents. Compare the responses of the agents. Evaluate the integration of the agents with the company's existing systems.

Comparing agent responses and testing system integration do not specify metrics for summarization accuracy or processing time. The option also does not describe how patient data or knowledge bases will be protected to satisfy the company's privacy policies. Its focus on multiple agents and integrations is less directly suited to a three-week accuracy-focused proof of concept. [3]

<https://docs.aws.amazon.com/bedrock/latest/userguide/knowledge-base.html>

Question 26

Single choice · Domain: D5 Testing, Validation, and Troubleshooting › Test GenAI applications

An ecommerce company is building an internal platform to develop generative AI applications by using Amazon Bedrock foundation models (FMs). Developers need to select models based on evaluations that are aligned to ecommerce use cases. The platform must display accuracy metrics for text generation and summarization in dashboards. The company has custom ecommerce datasets to use as standardized evaluation inputs.

Which combination of steps will meet these requirements with the LEAST operational overhead? (Select TWO.)

- A.** Import the datasets to an Amazon S3 bucket. Provide appropriate IAM permissions and cross-origin resource sharing (CORS) permissions to give the evaluation jobs access to the datasets.
- B.** Import the datasets to an Amazon S3 bucket. Provide appropriate IAM permissions and a VPC endpoint configuration to give the evaluation jobs access to the datasets.
- C.** Configure an AWS Lambda function to create model evaluation jobs on a schedule in the Amazon Bedrock console. Provide the URI of the S3 bucket that contains the datasets as an input. Configure the evaluation jobs to measure the real world knowledge (RWK) score for text generation and BERTScore for summarization. Configure a second Lambda function to check the status of the jobs and publish custom logs to Amazon CloudWatch. Create a custom Amazon CloudWatch Logs Insights dashboard.
- D.** Use Amazon SageMaker Clarify on a schedule to create model evaluation jobs. Use open source frameworks to create and run standardized evaluations. Publish results to Amazon CloudWatch namespaces. Use an AWS Lambda function to check the status of the jobs and publish custom logs to Amazon CloudWatch. Create a custom Amazon CloudWatch Logs Insights dashboard.
- E.** Run an Amazon SageMaker AI notebook job on a schedule by using the fmvelos or ragas framework to run evaluations that use the datasets in the S3 bucket. Write Python code in the notebook that makes direct InvokeModel API calls to the FMs and processes their responses for evaluation. Publish job status and results to Amazon CloudWatch Logs to measure the real world knowledge (RWK) score for text generation and toxicity for summarization as metrics for accuracy. Create a custom CloudWatch Logs Insights dashboard.

Answer

- **A. Import the datasets to an Amazon S3 bucket. Provide appropriate IAM permissions and cross- origin resource sharing (CORS) permissions to give the evaluation jobs access to the datasets.**
- **C. Configure an AWS Lambda function to create model evaluation jobs on a schedule in the Amazon Bedrock console. Provide the URI of the S3 bucket that contains the datasets as an input. Configure the evaluation jobs to measure the real world knowledge (RWK) score for text generation and BERTScore for summarization. Configure a second Lambda function to check the status of the jobs and publish custom logs to Amazon CloudWatch. Create a custom Amazon CloudWatch Logs Insights dashboard.**

Explanation

Importing the standardized evaluation datasets into S3 and granting the required IAM and CORS permissions lets Amazon Bedrock evaluation jobs access them. Using Bedrock model evaluation jobs with

real world knowledge (RWK) for text generation and BERTScore for summarization provides metrics aligned to the requested tasks, while Lambda and CloudWatch can automate job monitoring and dashboarding. This uses Bedrock's evaluation capabilities rather than requiring a custom evaluation pipeline.

https://docs.aws.amazon.com/us_en/bedrock/latest/userguide/model-evaluation-security-cors.html

<https://docs.aws.amazon.com/bedrock/latest/userguide/evaluation-data-management.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/model-evaluation-tasks-text-summary.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/model-evaluation-built-in-metrics.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/model-evaluation-metrics.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/model-evaluation-jobs-management-create.html>

✓ **A** Import the datasets to an Amazon S3 bucket. Provide appropriate IAM permissions and cross-origin resource sharing (CORS) permissions to give the evaluation jobs access to the datasets.

Amazon Bedrock model evaluation uses datasets in S3, and access requires the appropriate permissions and bucket configuration. CORS is relevant to enabling the evaluation workflow to access the S3 data; a VPC endpoint configuration is not the required substitute in this scenario.

https://docs.aws.amazon.com/us_en/bedrock/latest/userguide/model-evaluation-security-cors.html

<https://docs.aws.amazon.com/bedrock/latest/userguide/evaluation-data-management.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/model-evaluation-security-data.html>

✗ **B** Import the datasets to an Amazon S3 bucket. Provide appropriate IAM permissions and a VPC endpoint configuration to give the evaluation jobs access to the datasets.

S3 datasets and IAM permissions are appropriate, but this option substitutes a VPC endpoint configuration for the CORS permissions needed for the evaluation data access setup described here. Adding a VPC endpoint does not meet that requirement.

https://docs.aws.amazon.com/us_en/bedrock/latest/userguide/model-evaluation-security-cors.html

<https://docs.aws.amazon.com/bedrock/latest/userguide/evaluation-data-management.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/model-evaluation-security-data.html>

✓ **C** Configure an AWS Lambda function to create model evaluation jobs on a schedule in the Amazon Bedrock console. Provide the URI of the S3 bucket that contains the datasets as an input. Configure the evaluation jobs to measure the real world knowledge (RWK) score for text generation and BERTScore for summarization. Configure a second Lambda function to check the status of the jobs and publish custom logs to Amazon CloudWatch. Create a custom Amazon CloudWatch Logs Insights dashboard. Bedrock model evaluation jobs can use S3 evaluation data and provide task-appropriate metrics, including RWK for text generation and BERTScore for summarization. Lambda can automate job creation and status monitoring, with CloudWatch Logs Insights presenting the resulting metrics in dashboards, avoiding a separately built evaluation framework.

<https://docs.aws.amazon.com/bedrock/latest/userguide/model-evaluation-report.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/model-evaluation-jobs-management-list.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/model-evaluation-tasks-text-summary.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/model-evaluation-built-in-metrics.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/model-evaluation-metrics.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/model-evaluation-jobs-management-create.html>

✗ **D** Use Amazon SageMaker Clarify on a schedule to create model evaluation jobs. Use open source frameworks to create and run standardized evaluations. Publish results to Amazon CloudWatch namespaces. Use an AWS Lambda function to check the status of the jobs and publish custom logs to Amazon CloudWatch. Create a custom Amazon CloudWatch Logs Insights dashboard.

This approach relies on SageMaker Clarify and open-source evaluation frameworks instead of using Bedrock's purpose-built model evaluation jobs and metrics. It therefore requires more integration and operational work than the Bedrock-based approach that meets the requirements.

<https://docs.aws.amazon.com/bedrock/latest/userguide/model-evaluation-built-in-metrics.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/model-evaluation-jobs-management-create.html>

✗ **E** Run an Amazon SageMaker AI notebook job on a schedule by using the fmvelos or ragas framework to run evaluations that use the datasets in the S3 bucket. Write Python code in the notebook that makes direct InvokeModel API calls to the FMs and processes their responses for evaluation. Publish job status and results to Amazon CloudWatch Logs to measure the real world knowledge (RWK) score for text generation and toxicity for summarization as metrics for accuracy. Create a custom CloudWatch Logs Insights dashboard.

This option requires custom notebook code to invoke models, process responses, and calculate evaluations, which adds operational overhead compared with Bedrock evaluation jobs. It also uses toxicity for summarization rather than the requested accuracy-oriented summarization metric, BERTScore.

<https://docs.aws.amazon.com/bedrock/latest/userguide/model-evaluation-tasks-text-summary.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/model-evaluation-built-in-metrics.html>

<https://docs.aws.amazon.com/bedrock/latest/userguide/model-evaluation-metrics.html>

Dumpsgate