KaozhengPro

IT認證考試題庫

專業平臺

考證寶提供最新考古題與模擬試題
協助您高效通過認證考試

www.kaozhengpro.com

Exam : **AIP-C01**

Title : AWS Certified Generative AI Developer - Professional

Version : DEMO

1.A company provides a service that helps users from around the world discover new restaurants. The service has 50 million monthly active users. The company wants to implement a semantic search solution across a database that contains 20 million restaurants and 200 million reviews. The company currently stores the data in PostgreSQL.
The solution must support complex natural language queries and return results for at least 95% of queries within 500 ms. The solution must maintain data freshness for restaurant details that update hourly. The solution must also scale cost-effectively during peak usage periods.
Which solution will meet these requirements with the LEAST development effort?
A. Migrate the restaurant data to Amazon OpenSearch Service. Implement keyword-based search rules that use custom analyzers and relevance tuning to find restaurants based on attributes such as cuisine type, features, and location. Create Amazon API Gateway HTTP API endpoints to transform user queries into structured search parameters.
B. Migrate the restaurant data to Amazon OpenSearch Service. Use a foundation model (FM) in Amazon Bedrock to generate vector embeddings from restaurant descriptions, reviews, and menu items. When users submit natural language queries, convert the queries to embeddings by using the same FM. Perform k-nearest neighbors (k-NN) searches to find semantically similar results.
C. Keep the restaurant data in PostgreSQL and implement a pgvector extension. Use a foundation model (FM) in Amazon Bedrock to generate vector embeddings from restaurant data. Store the vector embeddings directly in PostgreSQL. Create an AWS Lambda function to convert natural language queries to vector representations by using the same FM. Configure the Lambda function to perform similarity searches within the database.
D. Migrate restaurant data to an Amazon Bedrock knowledge base by using a custom ingestion pipeline. Configure the knowledge base to automatically generate embeddings from restaurant information. Use the Amazon Bedrock Retrieve API with built-in vector search capabilities to query the knowledge base directly by using natural language input.
Answer: B
Explanation:
Option B best satisfies the requirements while minimizing development effort by combining managed semantic search capabilities with fully managed foundation models. AWS Generative AI guidance describes semantic search as a vector-based retrieval pattern where both documents and user queries are embedded into a shared vector space. Similarity search (such as k-nearest neighbors) then retrieves results based on meaning rather than exact keywords.
Amazon OpenSearch Service natively supports vector indexing and k-NN search at scale. This makes it well suited for large datasets such as 20 million restaurants and 200 million reviews while still achieving sub-second latency for the majority of queries. Because OpenSearch is a distributed, managed service, it automatically scales during peak traffic periods and provides cost-effective performance compared with building and tuning custom vector search pipelines on relational databases.
Using Amazon Bedrock to generate embeddings significantly reduces development complexity. AWS manages the foundation models, eliminates the need for custom model hosting, and ensures consistency by using the same FM for both document embeddings and query embeddings. This aligns directly with AWS-recommended semantic search architectures and removes the need for model lifecycle management.
Hourly updates to restaurant data can be handled efficiently through incremental re-indexing in OpenSearch without disrupting query performance. This approach cleanly separates transactional data

storage from search workloads, which is a best practice in AWS architectures.
Option A does not meet the semantic search requirement because keyword-based search cannot reliably interpret complex natural language intent.
Option C introduces scalability and performance risks by running large-scale vector similarity searches inside PostgreSQL, which increases operational complexity.
Option D adds unnecessary ingestion and abstraction layers intended for retrieval-augmented generation, not high-throughput semantic search.
Therefore, Option B provides the optimal balance of performance, scalability, data freshness, and minimal development effort using AWS Generative AI services.

2.A company is using Amazon Bedrock and Anthropic Claude 3 Haiku to develop an AI assistant. The AI assistant normally processes 10,000 requests each hour but experiences surges of up to 30,000 requests each hour during peak usage periods. The AI assistant must respond within 2 seconds while operating across multiple AWS Regions.
The company observes that during peak usage periods, the AI assistant experiences throughput bottlenecks that cause increased latency and occasional request timeouts. The company must resolve the performance issues.
Which solution will meet this requirement?
A. Purchase provisioned throughput and sufficient model units (MUs) in a single Region. Configure the application to retry failed requests with exponential backoff.
B. Implement token batching to reduce API overhead. Use cross-Region inference profiles to automatically distribute traffic across available Regions.
C. Set up auto scaling AWS Lambda functions in each Region. Implement client-side round-robin request distribution. Purchase one model unit (MU) of provisioned throughput as a backup.
D. Implement batch inference for all requests by using Amazon S3 buckets across multiple Regions. Use Amazon SQS to set up an asynchronous retrieval process.
Answer: B
Explanation:
Option B is the correct solution because it directly addresses both throughput bottlenecks and latency requirements using native Amazon Bedrock performance optimization features that are designed for real-time, high-volume generative AI workloads.
Amazon Bedrock supports cross-Region inference profiles, which allow applications to transparently route inference requests across multiple AWS Regions. During peak usage periods, traffic is automatically distributed to Regions with available capacity, reducing throttling, request queuing, and timeout risks. This approach aligns with AWS guidance for building highly available, low-latency GenAI applications that must scale elastically across geographic boundaries.
Token batching further improves efficiency by combining multiple inference requests into a single model invocation where applicable. AWS Generative AI documentation highlights batching as a key optimization technique to reduce per-request overhead, improve throughput, and better utilize model capacity. This is especially effective for lightweight, low-latency models such as Claude 3 Haiku, which are designed for fast responses and high request volumes.
Option A does not meet the requirement because purchasing provisioned throughput in a single Region creates a regional bottleneck and does not address multi-Region availability or traffic spikes beyond reserved capacity. Retries increase load and latency rather than resolving the root cause.

Option C improves application-layer scaling but does not solve model-side throughput limits. Client-side round-robin routing lacks awareness of real-time model capacity and can still send traffic to saturated Regions.
Option D is unsuitable because batch inference with asynchronous retrieval is designed for offline or non-interactive workloads. It cannot meet a strict 2-second response time requirement for an interactive AI assistant.
Therefore, Option B provides the most effective and AWS-aligned solution to achieve low latency, global scalability, and high throughput during peak usage periods.

3.A company uses an AI assistant application to summarize the company's website content and provide information to customers. The company plans to use Amazon Bedrock to give the application access to a foundation model (FM).
The company needs to deploy the AI assistant application to a development environment and a production environment. The solution must integrate the environments with the FM. The company wants to test the effectiveness of various FMs in each environment. The solution must provide product owners with the ability to easily switch between FMs for testing purposes in each environment.
Which solution will meet these requirements?
A. Create one AWS CDK application. Create multiple pipelines in AWS CodePipeline. Configure each pipeline to have its own settings for each FM. Configure the application to invoke the Amazon Bedrock FMs by using the aws_bedrock.ProvisionedModel.fromProvisionedModelArn() method.
B. Create a separate AWS CDK application for each environment. Configure the applications to invoke the Amazon Bedrock FMs by using the aws_bedrock.FoundationModel.fromFoundationModelId() method. Create a separate pipeline in AWS CodePipeline for each environment.
C. Create one AWS CDK application. Configure the application to invoke the Amazon Bedrock FMs by using the aws_bedrock.FoundationModel.fromFoundationModelId() method. Create a pipeline in AWS CodePipeline that has a deployment stage for each environment that uses AWS CodeBuild deploy actions.
D. Create one AWS CDK application for the production environment. Configure the application to invoke the Amazon Bedrock FMs by using the aws_bedrock.ProvisionedModel.fromProvisionedModelArn() method. Create a pipeline in AWS CodePipeline. Configure the pipeline to deploy to the production environment by using an AWS CodeBuild deploy action. For the development environment, manually recreate the resources by referring to the production application code.
Answer: C
Explanation:
Option C best satisfies the requirement for flexible FM testing across environments while minimizing operational complexity and aligning with AWS-recommended deployment practices. Amazon Bedrock supports invoking on-demand foundation models through the Foundation Model abstraction, which allows applications to dynamically reference different models without requiring dedicated provisioned capacity. This is ideal for experimentation and A/B testing in both development and production environments.
Using a single AWS CDK application ensures infrastructure consistency and reduces duplication. Environment-specific configuration, such as selecting different foundation model IDs, can be externalized through parameters, context variables, or environment-specific configuration files. This allows product

owners to easily switch between FMs in each environment without modifying application logic.
A single AWS Code Pipeline with distinct deployment stages for development and production is an AWS best practice for multi-environment deployments. It enforces consistent build and deployment steps while still allowing environment-level customization. AWS Code Build deploy actions enable automated, repeatable deployments, reducing manual errors and improving governance.
Option A increases complexity by introducing multiple pipelines and relies on provisioned models, which are not necessary for FM evaluation and experimentation. Provisioned throughput is better suited for predictable, high-volume production workloads rather than frequent model switching.
Option B creates unnecessary operational overhead by duplicating CDK applications and pipelines, making long-term maintenance more difficult.
Option D directly conflicts with infrastructure-as-code best practices by manually recreating development resources, which increases configuration drift and reduces reliability.
Therefore, Option C provides the most flexible, scalable, and AWS-aligned solution for testing and switching foundation models across development and production environments.

4.A company deploys multiple Amazon Bedrock–based generative AI (GenAI) applications across multiple business units for customer service, content generation, and document analysis. Some applications show unpredictable token consumption patterns. The company requires a comprehensive observability solution that provides real-time visibility into token usage patterns across multiple models. The observability solution must support custom dashboards for multiple stakeholder groups and provide alerting capabilities for token consumption across all the foundation models that the company's applications use.
Which combination of solutions will meet these requirements with the LEAST operational overhead? (Select TWO.)
A. Use Amazon CloudWatch metrics as data sources to create custom Amazon QuickSight dashboards that show token usage trends and usage patterns across FMs.
B. Use CloudWatch Logs Insights to analyze Amazon Bedrock invocation logs for token consumption patterns and usage attribution by application. Create custom queries to identify high-usage scenarios. Add log widgets to dashboards to enable continuous monitoring.
C. Create custom Amazon CloudWatch dashboards that combine native Amazon Bedrock token and invocation CloudWatch metrics. Set up CloudWatch alarms to monitor token usage thresholds.
D. Create dashboards that show token usage trends and patterns across the company's FMs by using an Amazon Bedrock zero-ETL integration with Amazon Managed Grafana.
E. Implement Amazon EventBridge rules to capture Amazon Bedrock model invocation events. Route token usage data to Amazon OpenSearch Serverless by using Amazon Data Firehose. Use OpenSearch dashboards to analyze usage patterns.
Answer: C, D
Explanation:
The combination of Options C and D delivers comprehensive, real-time observability for Amazon Bedrock workloads with the least operational overhead by relying on native integrations and managed services.
Amazon Bedrock publishes built-in CloudWatch metrics for model invocations and token usage. Option C leverages these native metrics directly, allowing teams to build centralized CloudWatch dashboards without additional data pipelines or custom processing. CloudWatch alarms provide

threshold-based alerting for token consumption, enabling proactive cost and usage control across all foundation models. This approach aligns with AWS guidance to use native service metrics whenever possible to reduce operational complexity.
Option D complements CloudWatch by enabling advanced, stakeholder-specific visualizations through Amazon Managed Grafana. The zero-ETL integration allows Bedrock and CloudWatch metrics to be visualized directly in Grafana without building ingestion pipelines or managing storage layers. Grafana dashboards are particularly well suited for serving different audiences, such as engineering, finance, and product teams, each with customized views of token usage and trends.
Option A introduces unnecessary complexity by adding a business intelligence layer that is better suited for historical analytics than real-time operational monitoring.
Option B is useful for deep log analysis but requires query maintenance and does not provide efficient real-time dashboards at scale.
Option E involves multiple services and custom data flows, significantly increasing operational overhead compared to native metric-based observability.
By combining CloudWatch dashboards and alarms with Managed Grafana’s zero-ETL visualization capabilities, the company achieves real-time visibility, flexible dashboards, and automated alerting across all Amazon Bedrock foundation models with minimal operational effort.

5.An enterprise application uses an Amazon Bedrock foundation model (FM) to process and analyze 50 to 200 pages of technical documents. Users are experiencing inconsistent responses and receiving truncated outputs when processing documents that exceed the FM's context window limits.
Which solution will resolve this problem?
A. Configure fixed-size chunking at 4,000 tokens for each chunk with 20% overlap. Use application-level logic to link multiple chunks sequentially until the FM's maximum context window of 200,000 tokens is reached before making inference calls.
B. Use hierarchical chunking with parent chunks of 8,000 tokens and child chunks of 2,000 tokens. Use Amazon Bedrock Knowledge Bases built-in retrieval to automatically select relevant parent chunks based on query context. Configure overlap tokens to maintain semantic continuity.
C. Use semantic chunking with a breakpoint percentile threshold of 95% and a buffer size of 3 sentences. Use the Retrieve And Generate API to dynamically select the most relevant chunks based on embedding similarity scores.
D. Create a pre-processing AWS Lambda function that analyzes document token count by using the FM's tokenizer. Configure the Lambda function to split documents into equal segments that fit within 80% of the context window. Configure the Lambda function to process each segment independently before aggregating the results.
Answer: C
Explanation:
Option C directly addresses the root cause of truncated and inconsistent responses by using AWS-recommended semantic chunking and dynamic retrieval rather than static or sequential chunk processing. Amazon Bedrock documentation emphasizes that foundation models have fixed context windows and that sending oversized or poorly structured input can lead to truncation, loss of context, and degraded output quality.
Semantic chunking breaks documents based on meaning instead of fixed token counts. By using a breakpoint percentile threshold and sentence buffers, the content remains coherent and semantically

complete. This approach reduces the likelihood that important concepts are split across chunks, which is a common cause of inconsistent summarization results.
The Retrieve And Generate API is designed specifically to handle large documents that exceed a model's context window. Instead of forcing all content into a single inference call, the API generates embeddings for chunks and dynamically selects only the most relevant chunks based on similarity to the user query. This ensures that the FM receives only high-value context while staying within its context window limits.
Option A is ineffective because chaining chunks sequentially does not align with how FMs process context and risks exceeding context limits or introducing irrelevant information.
Option B improves structure but still relies on larger parent chunks, which can lead to inefficiencies when processing very large documents.
Option D processes segments independently, which often causes loss of global context and inconsistent summaries.
Therefore, Option C is the most robust, AWS-aligned solution for resolving truncation and consistency issues when processing large technical documents with Amazon Bedrock.