AWS Certified Generative AI Developer - Professional (AIP-C01) Exam Guide

Introduction

The AWS Certified Generative AI Developer - Professional (AIP-C01) exam is intended for individuals who perform a GenAI developer role. The exam validates a candidate's ability to effectively integrate foundation models (FMs) into applications and business workflows. This certification demonstrates practical knowledge of how to implement GenAI solutions into production environments by using AWS technologies.

The exam also validates a candidate's ability to complete the following tasks:

- Design and implement solutions by using vector stores, Retrieval Augmented Generation (RAG), knowledge bases, and other GenAI architectures.
- Integrate FMs into applications and business workflows.
- Apply prompt engineering and management techniques.
- Implement agentic AI solutions.
- Optimize GenAI applications for cost, performance, and business value.
- Implement security, governance, and Responsible AI practices.
- Troubleshoot, monitor, and optimize GenAI applications.
- Evaluate FMs for quality and responsibility.

Target candidate description

The target candidate should have 2 or more years of experience building production-grade applications on AWS or with open-source technologies, general AI/ML or data engineering experience, and 1 year of hands-on experience implementing GenAI solutions.

Recommended AWS knowledge

The target candidate should have the following AWS knowledge:

- Experience with AWS compute, storage, and networking services
- Understanding of AWS security best practices and identity management
- Experience with AWS deployment and infrastructure as code (IaC) tools
- Familiarity with AWS monitoring and observability services

- Understanding of AWS cost optimization principles

Job tasks that are out of scope for the target candidate

The following list contains job tasks that the target candidate is not expected to be able to perform. This list is non-exhaustive. These tasks are out of scope for the exam:

- Model development and training
- Advanced ML techniques
- Data engineering and feature engineering

Refer to the Appendix for a list of technologies and concepts that might appear on the exam, a list of in-scope AWS services and features, and a list of out-of-scope AWS services and features.

Exam content

Question types

The exam contains one or more of the following question types:

- **Multiple choice:** Has one correct response and three incorrect responses (distractors).
- **Multiple response:** Has two or more correct responses out of five or more response options. You must select all the correct responses to receive credit for the question.
- **Ordering:** Has a list of 3–5 responses to complete a specified task. You must select the correct responses and place the responses in the correct order to receive credit for the question.
- **Matching:** Has a list of responses to match with a list of 3–7 prompts. You must match all the pairs correctly to receive credit for the question.

Unanswered questions are scored as incorrect. There is no penalty for guessing. The exam includes 65 questions that affect your score.[1]

Unscored content

The exam includes 10 unscored questions that do not affect your score. AWS collects information about performance on these unscored questions to evaluate them for future use as scored questions. The unscored questions are not identified on the exam.

Exam results

The AWS Certified Generative AI Developer - Professional (AIP-C01) exam has a pass or fail designation. The exam is scored against a minimum standard established by AWS professionals who follow certification industry best practices and guidelines.

Your results for the exam are reported as a scaled score of 100–1,000. The minimum passing score is 750. Your score shows how you performed on the exam as a whole and whether you passed. Scaled scoring models help equate scores across multiple exam forms that might have slightly different difficulty levels.

Your score report could contain a table of classifications of your performance at each section level. The exam uses a compensatory scoring model, which means that you do not need to achieve a passing score in each section. You need to pass only the overall exam.

Each section of the exam has a specific weighting, so some sections have more questions than other sections have. The table of classifications contains general information that highlights your strengths and weaknesses. Use caution when you interpret section-level feedback.

[1] Does not apply to the beta version of the exam. You can find more information about beta exams in general on the AWS Certification website.

Content outline

This exam guide includes weightings, content domains, tasks, and skills for the exam. This guide does not provide a comprehensive list of the content on the exam.

The exam has the following content domains and weightings:

- Content Domain 1: Foundation Model Integration, Data Management, and Compliance (31% of scored content)
- Content Domain 2: Implementation and Integration (26% of scored content)
- Content Domain 3: AI Safety, Security, and Governance (20% of scored content)
- Content Domain 4: Operational Efficiency and Optimization for GenAI Applications (12% of scored content)
- Content Domain 5: Testing, Validation, and Troubleshooting (11% of scored content)

Content Domain 1: Foundation Model Integration, Data Management, and Compliance

Task 1.1: Analyze requirements and design GenAI solutions.

Skill 1.1.1: Create comprehensive architectural designs that align with specific business needs and technical constraints (for example, by using appropriate FMs, integration patterns, deployment strategies).
Skill 1.1.2: Develop technical proof-of-concept implementations to validate feasibility, performance characteristics, and business value before proceeding to full-scale deployment (for example, by using Amazon Bedrock).
Skill 1.1.3: Create standardized technical components to ensure consistent implementation across multiple deployment scenarios (for example, by using the AWS Well-Architected Framework, AWS WA Tool Generative AI Lens).

Task 1.2: Select and configure FMs.

Skill 1.2.1: Assess and choose FMs to ensure optimal alignment with specific business use cases and technical requirements (for example, by using performance benchmarks, capability analysis, limitation evaluation).
Skill 1.2.2: Create flexible architecture patterns to enable dynamic model selection and provider switching without requiring code modifications (for example, by using AWS Lambda, Amazon API Gateway, AWS AppConfig).
Skill 1.2.3: Design resilient AI systems to ensure continuous operation during service disruptions (for example, by using AWS Step Functions circuit breaker patterns, Amazon Bedrock Cross-Region Inference for models that have limited regional availability, cross-Region model deployment, graceful degradation strategies).
Skill 1.2.4: Implement FM customization deployment and lifecycle management (for example, by using Amazon SageMaker AI to deploy domain-specific fine-tuned models, parameter-efficient adaptation techniques such as low-rank adaptation [LoRA] and adapters for model deployment, SageMaker Model Registry for versioning and to deploy customized models, automated deployment pipelines to update models, rollback strategies for failed deployments, lifecycle management to retire and replace models).

Task 1.3: Implement data validation and processing pipelines for FM consumption.

Skill 1.3.1: Create comprehensive data validation workflows to ensure data meets quality standards for FM consumption (for example, by using AWS Glue Data Quality, SageMaker Data Wrangler, custom Lambda functions, Amazon CloudWatch metrics).
Skill 1.3.2: Create data processing workflows to handle complex data types including text, image, audio, and tabular data with specialized processing requirements for FM consumption (for example, by using Amazon Bedrock multimodal models, SageMaker Processing, AWS Transcribe, advanced multimodal pipeline architectures).
Skill 1.3.3: Format input data for FM inference according to model-specific requirements (for example, by using JSON formatting for Amazon Bedrock API requests, structured data preparation for SageMaker AI endpoints, conversation formatting for dialog-based applications).

Skill 1.3.4: Enhance input data quality to improve FM response quality and consistency (for example, by using Amazon Bedrock to reformat text, Amazon Comprehend to extract entities, Lambda functions to normalize data).

Task 1.4: Design and implement vector store solutions.

Skill 1.4.1: Create advanced vector database architectures specifically for FM augmentation to enable efficient semantic retrieval beyond traditional search capabilities (for example, by using Amazon Bedrock Knowledge Bases for hierarchical organization, Amazon OpenSearch Service with the Neural plugin for Amazon Bedrock integration for topic-based segmentation, Amazon RDS with Amazon S3 document repositories, Amazon DynamoDB with vector databases for metadata and embeddings).
Skill 1.4.2: Develop comprehensive metadata frameworks to improve search precision and context awareness for FM interactions (for example, by using S3 object metadata for document timestamps, custom attributes for authorship information, tagging systems for domain classification).
Skill 1.4.3: Implement high-performance vector database architectures to optimize semantic search performance at scale for FM retrieval (for example, by using OpenSearch sharding strategies, multi-index approaches for specialized domains, hierarchical indexing techniques).
Skill 1.4.4: Use AWS services to create integration components to connect with resources (for example, document management systems, knowledge bases, internal wikis for comprehensive data integration in GenAI applications).
Skill 1.4.5: Design and deploy data maintenance systems to ensure that vector stores contain current and accurate information for FM augmentation (for example, by using incremental update mechanisms, real-time change detection systems, automated synchronization workflows, scheduled refresh pipelines).

Task 1.5: Design retrieval mechanisms for FM augmentation.

Skill 1.5.1: Develop effective document segmentation approaches to optimize retrieval performance for FM context augmentation (for example, by using Amazon Bedrock chunking capabilities, Lambda functions to implement fixed-size chunking, custom processing for hierarchical chunking based on content structure).

Skill 1.5.2: Select and configure optimal embedding solutions to create efficient vector representations for semantic search (for example, by using Amazon Titan embeddings based on dimensionality and domain fit, by evaluating performance characteristics of Amazon Bedrock embedding models, by using Lambda functions to batch generate embeddings).
Skill 1.5.3: Deploy and configure vector search solutions to enable semantic search capabilities for FM augmentation (for example, by using OpenSearch Service with vector search capabilities, Amazon Aurora with the pgvector extension, Amazon Bedrock Knowledge Bases with managed vector store functionality).
Skill 1.5.4: Create advanced search architectures to improve the relevance and accuracy of retrieved information for FM context (for example, by using OpenSearch for semantic search, hybrid search that combines keywords and vectors, Amazon Bedrock reranker models).
Skill 1.5.5: Develop sophisticated query handling systems to improve the retrieval effectiveness and result quality for FM augmentation (for example, by using Amazon Bedrock for query expansion, Lambda functions for query decomposition, Step Functions for query transformation).
Skill 1.5.6: Create consistent access mechanisms to enable seamless integration with FMs (for example, by using function calling interfaces for vector search, Model Context Protocol [MCP] clients for vector queries, standardized API patterns for retrieval augmentation).

Task 1.6: Implement prompt engineering strategies and governance for FM interactions.

Skill 1.6.1: Create effective model instruction frameworks to control FM behavior and outputs (for example, by using Amazon Bedrock Prompt Management to enforce role definitions, Amazon Bedrock Guardrails to enforce responsible AI guidelines, template configurations to format responses).
Skill 1.6.2: Build interactive AI systems to maintain context and improve user interactions with FMs (for example, by using Step Functions for clarification workflows, Amazon Comprehend for intent recognition, DynamoDB for conversation history storage).

Skill 1.6.3: Implement comprehensive prompt management and governance systems to ensure consistency and oversight of FM operations (for example, by using Amazon Bedrock Prompt Management to create parameterized templates and approval workflows, Amazon S3 to store template repositories, AWS CloudTrail to track usage, Amazon CloudWatch Logs to log access).
Skill 1.6.4: Develop quality assurance systems to ensure prompt effectiveness and reliability for FMs (for example, by using Lambda functions to verify expected output, Step Functions to test edge cases, CloudWatch to test prompt regression).
Skill 1.6.5: Enhance FM performance to refine prompts iteratively and improve response quality beyond basic prompting techniques (for example, by using structured input components, output format specifications, chain-of-thought instruction patterns, feedback loops).
Skill 1.6.6: Design complex prompt systems to handle sophisticated tasks with FMs (for example, by using Amazon Bedrock Prompt Flows for sequential prompt chains, conditional branching based on model responses, reusable prompt components, integrated pre-processing and post-processing steps).

Content Domain 2: Implementation and Integration

Task 2.1: Implement agentic AI solutions and tool integrations.

Skill 2.1.1: Develop intelligent autonomous systems with appropriate memory and state management capabilities (for example, by using Strands Agents and AWS Agent Squad for multi-agent systems, MCP for agent-tool interactions).
Skill 2.1.2: Create advanced problem-solving systems to give FMs the ability to break down and solve complex problems by following structured reasoning steps (for example, by using Step Functions to implement ReAct patterns and chain-of-thought reasoning approaches).
Skill 2.1.3: Develop safeguarded AI workflows to ensure controlled FM behavior (for example, by using Step Functions to implement stopping conditions, Lambda functions to implement timeout mechanisms, IAM policies to enforce resource boundaries, circuit breakers to mitigate failures).

Skill 2.1.4: Create sophisticated model coordination systems to optimize performance across multiple capabilities (for example, by using specialized FMs to perform complex tasks, custom aggregation logic for model ensembles, model selection frameworks).
Skill 2.1.5: Develop collaborative AI systems to enhance FM capabilities with human expertise (for example, by using Step Functions to orchestrate review and approval processes, API Gateway to implement feedback collection mechanisms, human augmentation patterns).
Skill 2.1.6: Implement intelligent tool integrations to extend FM capabilities and to ensure reliable tool operations (for example, by using the Strands API to implement custom behaviors, standardized function definitions, Lambda functions to implement error handling and parameter validation).
Skill 2.1.7: Develop model extension frameworks to enhance FM capabilities (for example, by using Lambda functions to implement stateless MCP servers that provide lightweight tool access, Amazon ECS to implement MCP servers that provide complex tools, MCP client libraries to ensure consistent access patterns).

Task 2.2: Implement model deployment strategies.

Skill 2.2.1: Deploy FMs based on specific application needs and performance requirements (for example, by using Lambda functions for on-demand invocation, Amazon Bedrock provisioned throughput configurations, SageMaker AI endpoints to implement hybrid solutions).
Skill 2.2.2: Deploy FM solutions by addressing unique challenges of large language models (LLMs) that differ from traditional ML deployments (for example, by implementing container-based deployment patterns that are optimized for memory requirements, GPU utilization, and token processing capacity, by following specialized model loading strategies).
Skill 2.2.3: Develop optimized FM deployment approaches to balance performance and resource requirements for GenAI workloads (for example, by selecting appropriate models, by using smaller pre-trained models for specific tasks, by using API-based model cascading to perform routine queries).

Task 2.3: Design and implement enterprise integration architectures.

Skill 2.3.1: Create enterprise connectivity solutions to seamlessly incorporate FM capabilities into existing enterprise environments (for example, by using API-based

integrations with legacy systems, event-driven architectures to implement loose coupling, data synchronization patterns).
Skill 2.3.2: Develop integrated AI capabilities to enhance existing applications with GenAI functionality (for example, by using API Gateway to implement microservice integrations, Lambda functions for webhook handlers, Amazon EventBridge to implement event-driven integrations).
Skill 2.3.3: Create secure access frameworks to ensure appropriate security controls (for example, by using identity federation between FM services and enterprise systems, role-based access control for model and data access, least privilege API access to FMs).
Skill 2.3.4: Develop cross-environment AI solutions to ensure data compliance across jurisdictions while enabling FM access (for example, by using AWS Outposts for on-premises data integration, AWS Wavelength to perform edge deployments, secure routing between cloud and on-premises resources).
Skill 2.3.5: Implement CI/CD pipelines and GenAI gateway architectures to implement secure and compliant consumption patterns in enterprise environments (for example, by using AWS CodePipeline, AWS CodeBuild, automated testing frameworks for continuous deployment and testing of GenAI components with security scans and rollback support, centralized abstraction layers, observability and control mechanisms).

Task 2.4: Implement FM API integrations.

Skill 2.4.1: Create flexible model interaction systems (for example, by using Amazon Bedrock APIs to manage synchronous requests from various compute environments, language-specific AWS SDKs and Amazon SQS for asynchronous processing, API Gateway to provide custom API clients with request validation).
Skill 2.4.2: Develop real-time AI interaction systems to provide immediate feedback from FM (for example, by using Amazon Bedrock streaming APIs for incremental response delivery, WebSockets or server-sent events to generate text in real time, API Gateway to implement chunked transfer encoding).
Skill 2.4.3: Create resilient FM systems to ensure reliable operations (for example, by using the AWS SDK for exponential backoff, API Gateway to manage rate limiting, fallback mechanisms for graceful degradation, AWS X-Ray to provide observability across service boundaries).

Skill 2.4.4: Develop intelligent model routing systems to optimize model selection (for example, by using application code to implement static routing configurations, Step Functions for dynamic content-based routing to specialized FMs, intelligent model routing based on metrics, API Gateway with request transformations for routing logic).

Task 2.5: Implement application integration patterns and development tools.

Skill 2.5.1: Create FM API interfaces to address the specific requirements of GenAI workloads (for example, by using API Gateway to handle streaming responses, token limit management, retry strategies to handle model timeouts).
Skill 2.5.2: Develop accessible AI interfaces to accelerate adoption and integration of FMs (for example, by using AWS Amplify to develop declarative UI components, OpenAPI specifications for API-first development approaches, Amazon Bedrock Prompt Flows for no-code workflow builders).
Skill 2.5.3: Create business system enhancements (for example, by using Lambda functions to implement customer relationship management [CRM] enhancements, Step Functions to orchestrate document processing systems, Amazon Q Business data sources to provide internal knowledge tools, Amazon Bedrock Data Automation to manage automated data processing workflows).
Skill 2.5.4: Enhance developer productivity to accelerate development workflows for GenAI applications (for example, by using Amazon Q Developer to generate and refactor code, code suggestions for API assistance, AI component testing, performance optimization).
Skill 2.5.5: Develop advanced GenAI applications to implement sophisticated AI capabilities (for example, by using Strands Agents and AWS Agent Squad for AWS native orchestration, Step Functions to orchestrate agent design patterns, Amazon Bedrock to manage prompt chaining patterns).
Skill 2.5.6: Improve troubleshooting efficiency for FM applications (for example, by using CloudWatch Logs Insights to analyze prompts and responses, X-Ray to trace FM API calls, Amazon Q Developer to implement GenAI-specific error pattern recognition).

Content Domain 3: AI Safety, Security, and Governance

Task 3.1: Implement input and output safety controls.

Skill 3.1.1: Develop comprehensive content safety systems to protect against harmful user inputs to FMs (for example, by using Amazon Bedrock guardrails to filter content, Step Functions and Lambda functions to implement custom moderation workflows, real-time validation mechanisms).
Skill 3.1.2: Create content safety frameworks to prevent harmful outputs (for example, by using Amazon Bedrock guardrails to filter responses, specialized FM evaluations for content moderation and toxicity detection, text-to-SQL transformations to ensure deterministic results).
Skill 3.1.3: Develop accuracy verification systems to reduce hallucinations in FM responses (for example, by using Amazon Bedrock Knowledge Base to ground responses and perform fact-checking, confidence scoring and semantic similarity search for verification, JSON Schema to enforce structured outputs).
Skill 3.1.4: Create defense-in-depth safety systems to provide comprehensive protection against FM misuse (for example, by using Amazon Comprehend to develop pre-processing filters, Amazon Bedrock to implement model-based guardrails, Lambda functions to perform post-processing validation, API Gateway to implement API response filtering).
Skill 3.1.5: Implement advanced threat detection to protect against adversarial inputs and security vulnerabilities (for example, by using prompt injection and jailbreak detection mechanisms, input sanitization and content filters, safety classifiers, automated adversarial testing workflows).

Task 3.2: Implement data security and privacy controls.

Skill 3.2.1: Develop protected AI environments to ensure comprehensive security for FM deployments (for example, by using VPC endpoints to isolate networks, IAM policies to enforce secure data access patterns, AWS Lake Formation to provide granular data access, CloudWatch to monitor data access).
Skill 3.2.2: Develop privacy-preserving systems to protect sensitive information during FM interactions (for example, by using Amazon Comprehend and Amazon Macie to detect personally identifiable information [PII], Amazon Bedrock native data privacy features, Amazon Bedrock guardrails to filter outputs, Amazon S3 Lifecycle configurations to implement data retention policies).

Skill 3.2.3: Create privacy-focused AI systems to protect user privacy while maintaining FM utility and effectiveness (for example, by using data masking techniques, Amazon Comprehend PII detection, anonymization strategies for sensitive information, Amazon Bedrock guardrails).

Task 3.3: Implement AI governance and compliance mechanisms.

Skill 3.3.1: Develop compliance frameworks to ensure regulatory compliance for FM deployments (for example, by using SageMaker AI to develop programmatic model cards, AWS Glue to automatically track data lineage, metadata tagging for systematic data source attribution, CloudWatch Logs to collect comprehensive decision logs).
Skill 3.3.2: Implement data source tracking to maintain traceability in GenAI applications (for example, by using AWS Glue Data Catalog to register data sources, metadata tagging for source attribution in FM-generated content, CloudTrail for audit logging).
Skill 3.3.3: Create organizational governance systems to ensure consistent oversight of FM implementations (for example, by using comprehensive frameworks that align with organizational policies, regulatory requirements, and responsible AI principles).
Skill 3.3.4: Implement continuous monitoring and advanced governance controls to support safety audits and regulatory readiness (for example, by using automated detection for misuse, drift, and policy violations, bias drift monitoring, automated alerting and remediation workflows, token-level redaction, response logging, AI output policy filters).

Task 3.4: Implement responsible AI principles.

Skill 3.4.1: Develop transparent AI systems in FM outputs (for example, by using reasoning displays to provide user-facing explanations, CloudWatch to collect confidence metrics and quantify uncertainty, evidence presentation for source attribution, Amazon Bedrock agent tracing to provide reasoning traces).
Skill 3.4.2: Apply fairness evaluations to ensure unbiased FM outputs (for example, by using pre-defined fairness metrics in CloudWatch, Amazon Bedrock Prompt Management and Amazon Bedrock Prompt Flows to perform systematic A/B testing, Amazon Bedrock with LLM-as-a-judge solutions to perform automated model evaluations).

Skill 3.4.3: Develop policy-compliant AI systems to ensure adherence to responsible AI practices (for example, by using Amazon Bedrock guardrails based on policy requirements, model cards to document FM limitations, Lambda functions to perform automated compliance checks).

Content Domain 4: Operational Efficiency and Optimization for GenAI Applications

Task 4.1: Implement cost optimization and resource efficiency strategies.

Skill 4.1.1: Develop token efficiency systems to reduce FM costs while maintaining effectiveness (for example, by using token estimation and tracking, context window optimization, response size controls, prompt compression, context pruning, response limiting).
Skill 4.1.2: Create cost-effective model selection frameworks (for example, by using cost-capability tradeoff evaluation, tiered FM usage based on query complexity, inference cost balancing against response quality, price-to-performance ratio measurement, efficient inference patterns).
Skill 4.1.3: Develop high-performance FM systems to maximize resource utilization and throughput for GenAI workloads (for example, by using batching strategies, capacity planning, utilization monitoring, auto-scaling configurations, provisioned throughput optimization).
Skill 4.1.4: Create intelligent caching systems to reduce costs and improve response times by avoiding unnecessary FM invocations (for example, by using semantic caching, result fingerprinting, edge caching, deterministic request hashing, prompt caching).

Task 4.2: Optimize application performance.

Skill 4.2.1: Create responsive AI systems to address latency-cost tradeoffs and improve the user experience with FMs (for example, by using pre-computation to perform predictable queries, latency-optimized Amazon Bedrock models for time-sensitive applications, parallel requests for complex workflows, response streaming, performance benchmarking).

Skill 4.2.2: Enhance retrieval performance to improve the relevance and speed of retrieved information for FM context augmentation (for example, by using index optimization, query preprocessing, hybrid search implementation with custom scoring).
Skill 4.2.3: Implement FM throughput optimization to address the specific throughput challenges of GenAI workloads (for example, by using token processing optimization, batch inference strategies, concurrent model invocation management).
Skill 4.2.4: Enhance FM performance to achieve optimal results for specific GenAI use cases (for example, by using model-specific parameter configurations, A/B testing to evaluate improvements, appropriate temperature and top-k/top-p selection based on requirements).
Skill 4.2.5: Create efficient resource allocation systems specifically for FM workloads (for example, by using capacity planning for token processing requirements, utilization monitoring for prompt and completion patterns, auto-scaling configurations that are optimized for GenAI traffic patterns).
Skill 4.2.6: Optimize FM system performance for GenAI workflows (for example, by using API call profiling for prompt-completion patterns, vector database query optimization for retrieval augmentation, latency reduction techniques specific to LLM inference, efficient service communication patterns).

Task 4.3: Implement monitoring systems for GenAI applications.

Skill 4.3.1: Create holistic observability systems to provide complete visibility into FM application performance (for example, by using operational metrics, performance tracing, FM interaction tracing, business impact metrics with custom dashboards).
Skill 4.3.2: Implement comprehensive GenAI monitoring systems to proactively identify issues and evaluate key performance indicators specific to FM implementations (for example, by using CloudWatch to track token usage; prompt effectiveness; hallucination rates; and response quality, anomaly detection for token burst patterns and response drift, Amazon Bedrock Model Invocation Logs to perform detailed request and response analysis, performance benchmarks, cost anomaly detection).

Skill 4.3.3: Develop integrated observability solutions to provide actionable insights for FM applications (for example, by using operational metric dashboards, business impact visualizations, compliance monitoring, forensic traceability and audit logging, user interaction tracking, model behavior pattern tracking).
Skill 4.3.4: Create tool performance frameworks to ensure optimal tool operation and utilization for FMs (for example, by using call pattern tracking, performance metric collection, tool calling observability and multi-agent coordination tracking, usage baselines for anomaly detection).
Skill 4.3.5: Create vector store operational management systems to ensure optimal vector store operation and reliability for FM augmentation (for example, by using performance monitoring for vector databases, automated index optimization routines, data quality validation processes).
Skill 4.3.6: Develop FM-specific troubleshooting frameworks to identify unique GenAI failure modes that are not present in traditional ML systems (for example, by using golden datasets to detect hallucinations, output diffing techniques to conduct response consistency analysis, reasoning path tracing to identify logical errors, specialized observability pipelines).

Content Domain 5: Testing, Validation, and Troubleshooting

Task 5.1: Implement evaluation systems for GenAI.

Skill 5.1.1: Develop comprehensive assessment frameworks to evaluate the quality and effectiveness of FM outputs beyond traditional ML evaluation approaches (for example, by using metrics for relevance, factual accuracy, consistency, and fluency).
Skill 5.1.2: Create systematic model evaluation systems to identify optimal configurations (for example, by using Amazon Bedrock Model Evaluations, A/B testing and canary testing of FMs, multi-model evaluation, cost-performance analysis to measure token efficiency, latency-to-quality ratios, and business outcomes).
Skill 5.1.3: Develop user-centered evaluation mechanisms to continuously improve FM performance based on user experience (for example, by using feedback interfaces, rating systems for model outputs, annotation workflows to assess response quality).

Skill 5.1.4: Create systematic quality assurance processes to maintain consistent performance standards for FMs (for example, by using continuous evaluation workflows, regression testing for model outputs, automated quality gates for deployments).
Skill 5.1.5: Develop comprehensive assessment systems to ensure thorough evaluation from multiple perspectives for FM outputs (for example, by using RAG evaluation, automated quality assessment with LLM-as-a-Judge techniques, human feedback collection interfaces).
Skill 5.1.6: Implement retrieval quality testing to evaluate and optimize information retrieval components for FM augmentation (for example, by using relevance scoring, context matching verification, retrieval latency measurements).
Skill 5.1.7: Develop agent performance frameworks to ensure that agents perform tasks correctly and efficiently (for example, by using task completion rate measurements, tool usage effectiveness evaluations, Amazon Bedrock Agent evaluations, reasoning quality assessment in multi-step workflows).
Skill 5.1.8: Create comprehensive reporting systems to communicate performance metrics and insights effectively to stakeholders for FM implementations (for example, by using visualization tools, automated reporting mechanisms, model comparison visualizations).
Skill 5.1.9: Create deployment validation systems to maintain reliability during FM updates (for example, by using synthetic user workflows, AI-specific output validation for hallucination rates and semantic drift, automated quality checks to ensure response consistency).

Task 5.2: Troubleshoot GenAI applications.

Skill 5.2.1: Resolve content handling issues to ensure that necessary information is processed completely in FM interactions (for example, by using context window overflow diagnostics, dynamic chunking strategies, prompt design optimization, truncation-related error analysis).
Skill 5.2.2: Diagnose and resolve FM integration issues to identify and fix API integration problems specific to GenAI services (for example, by using error logging, request validation, response analysis).
Skill 5.2.3: Troubleshoot prompt engineering problems to improve FM response quality and consistency beyond basic prompt adjustments (for example, by using prompt testing frameworks, version comparison, systematic refinement).

Skill 5.2.4: Troubleshoot retrieval system issues to identify and resolve problems that affect information retrieval effectiveness for FM augmentation (for example, by using model response relevance analysis, embedding quality diagnostics, drift monitoring, vectorization issue resolution, chunking and preprocessing remediation, vector search performance optimization).
Skill 5.2.5: Troubleshoot prompt maintenance issues to continuously improve the performance of FM interactions (for example, by using template testing and CloudWatch Logs to diagnose prompt confusion, X-Ray to implement prompt observability pipelines, schema validation to detect format inconsistencies, systematic prompt refinement workflows).

Appendix

Technologies and concepts that might appear on the exam

The following list contains technologies and concepts that might appear on the exam. This list is non-exhaustive and is subject to change. The order and placement of the items in this list is no indication of their relative weight or importance on the exam:

- Retrieval Augmented Generation (RAG)
- Vector databases and embeddings
- Prompt engineering and management
- Foundation model (FM) integration
- Agentic AI systems
- Responsible AI practices
- Content safety and moderation
- Model evaluation and validation
- Cost optimization for AI workloads
- Performance tuning for AI applications
- Monitoring and observability for AI systems
- Security and governance for AI applications
- API design and integration patterns
- Event-driven architectures
- Serverless computing
- Container orchestration
- Infrastructure as code (IaC)

- CI/CD for AI applications
- Hybrid cloud architectures
- Enterprise system integration

Mentions of AWS services on the exam

AWS Certification is reducing the reading load on this exam by using official short names of well-known AWS service names that contain abbreviations or parenthetical information. For example, *Amazon Simple Notification Service (Amazon SNS)* appears on the exam as *Amazon SNS*.

The Help feature in the exam (available for every question) contains the list of the short AWS service names and their corresponding full names.

You can consult AWS Service Names on the AWS Certification website for the list of services that appear as their short names on the exam. Any services that are on the list but that are out of scope for the exam will not appear on the exam.

Note: Not every abbreviation is fully spelled out on the exam or available in the Help feature. The official full name for some AWS services includes an abbreviation that is never expanded (for example, Amazon API Gateway, Amazon EMR). The exam also might contain other abbreviations that the target audience is expected to know.

In-scope AWS services and features

The following list contains AWS services and features that are in scope for the exam. This list is non-exhaustive and is subject to change. AWS offerings appear in categories that align with the offerings' primary functions:

Analytics:

- Amazon Athena
- Amazon EMR
- AWS Glue
- Amazon Kinesis
- Amazon OpenSearch Service
- Amazon QuickSight
- Amazon Managed Streaming for Apache Kafka (Amazon MSK)

Application Integration:

- Amazon AppFlow
- AWS AppConfig
- Amazon EventBridge
- Amazon SNS
- Amazon SQS
- AWS Step Functions

Compute:

- AWS App Runner
- Amazon EC2
- AWS Lambda
- AWS Lambda@Edge
- AWS Outposts
- AWS Wavelength

Containers:

- Amazon ECR
- Amazon ECS
- Amazon EKS
- AWS Fargate

Customer Engagement:

- Amazon Connect

Database:

- Amazon Aurora
- Amazon DocumentDB
- Amazon DynamoDB
- Amazon DynamoDB Streams
- Amazon ElastiCache
- Amazon Neptune
- Amazon RDS

Developer Tools:

- AWS Amplify
- AWS CDK
- AWS CLI
- AWS CloudFormation
- AWS CodeArtifact
- AWS CodeBuild
- AWS CodeDeploy
- AWS CodePipeline
- AWS Tools and SDKs
- AWS X-Ray

Machine Learning:

- Amazon Augmented AI
- Amazon Bedrock
- Amazon Bedrock AgentCore
- Amazon Bedrock Knowledge Bases
- Amazon Bedrock Prompt Management
- Amazon Bedrock Prompt Flows
- Amazon Comprehend
- Amazon Kendra
- Amazon Lex
- Amazon Q Business
- Amazon Q Business Apps
- Amazon Q Developer
- Amazon Rekognition
- Amazon SageMaker AI
- Amazon SageMaker Clarify
- Amazon SageMaker Data Wrangler
- Amazon SageMaker Ground Truth
- Amazon SageMaker JumpStart
- Amazon SageMaker Model Monitor
- Amazon SageMaker Model Registry
- Amazon SageMaker Neo

- Amazon SageMaker Processing
- Amazon SageMaker Unified Studio
- Amazon Textract
- Amazon Titan
- Amazon Transcribe

Management and Governance:

- AWS Auto Scaling
- AWS Chatbot
- AWS CloudTrail
- Amazon CloudWatch
- Amazon CloudWatch Logs
- Amazon CloudWatch Synthetics
- AWS Cost Anomaly Detection
- AWS Cost Explorer
- Amazon Managed Grafana
- AWS Service Catalog
- AWS Systems Manager
- AWS Well-Architected Tool

Migration and Transfer:

- AWS DataSync
- AWS Transfer Family

Networking and Content Delivery:

- Amazon API Gateway
- AWS AppSync
- Amazon CloudFront
- Elastic Load Balancing (ELB)
- AWS Global Accelerator
- AWS PrivateLink
- Amazon Route 53
- Amazon VPC

Security, Identity, and Compliance:

- Amazon Cognito
- AWS Encryption SDK
- IAM
- IAM Access Analyzer
- IAM Identity Center
- AWS KMS
- Amazon Macie
- AWS Secrets Manager
- AWS WAF

Storage:

- Amazon EBS
- Amazon EFS
- Amazon S3
- Amazon S3 Intelligent-Tiering
- Amazon S3 Lifecycle policies
- Amazon S3 Cross-Region Replication

Out-of-scope AWS services and features

The following list contains AWS services and features that are out of scope for the exam. This list is non-exhaustive and is subject to change. AWS offerings that are entirely unrelated to the target job roles for the exam are excluded from this list:

Application Integration:

- Amazon MQ

Analytics:

- AWS Clean Rooms
- AWS Data Exchange
- Amazon DataZone
- Amazon FinSpace

Blockchain:

- Amazon Managed Blockchain (AMB)

Business Applications:

- Alexa for Business
- Amazon Chime
- AWS Wickr
- Amazon WorkDocs
- Amazon WorkMail

Cloud Financial Management:

- AWS Budgets
- AWS Cost and Usage Report
- Reserved Instance reports
- AWS Savings Plans

Compute:

- AWS Batch
- Amazon EC2 Image Builder
- Amazon ECS Anywhere
- Amazon EKS Anywhere
- AWS Elastic Beanstalk
- Amazon Lightsail
- AWS Local Zones
- AWS Serverless Application Repository

Containers:

- AWS App2Container
- AWS Copilot
- Red Hat OpenShift Service on AWS (ROSA)

Customer Engagement:

- Amazon SES

Database:

- Amazon Keyspaces
- Amazon Quantum Ledger Database (Amazon QLDB)
- Amazon Redshift
- Amazon Timestream

Developer Tools:

- AWS Cloud9
- AWS CloudShell
- Amazon CodeGuru
- AWS CodeStar
- Amazon Corretto

End User Computing:

- Amazon AppStream 2.0
- Amazon WorkLink
- Amazon WorkSpaces
- Amazon WorkSpaces Web

Frontend Web and Mobile:

- AWS Device Farm
- Amazon Location Service
- Amazon Pinpoint

Game Development:

- Amazon GameLift
- Amazon Lumberyard

Internet of Things (IoT):

- AWS IoT 1-Click
- AWS IoT Analytics
- AWS IoT Button
- AWS IoT Core
- AWS IoT Device Defender

- AWS IoT Device Management
- AWS IoT Events
- AWS IoT FleetWise
- AWS IoT Greengrass
- AWS IoT SiteWise
- AWS IoT TwinMaker

Management and Governance:

- AWS Console Mobile Application
- AWS Health Dashboard
- AWS License Manager
- AWS Proton
- AWS Trusted Advisor

Machine Learning:

- AWS DeepComposer
- AWS DeepRacer
- Amazon DevOps Guru
- Amazon Forecast
- Amazon Fraud Detector
- Amazon HealthLake
- Amazon Lookout for Equipment
- Amazon Lookout for Metrics
- Amazon Lookout for Vision
- Amazon Monitron
- AWS Panorama

Media Services:

- Amazon Elastic Transcoder
- AWS Elemental MediaConnect
- AWS Elemental MediaConvert
- AWS Elemental MediaLive
- AWS Elemental MediaPackage
- AWS Elemental MediaStore

- AWS Elemental MediaTailor
- Amazon Interactive Video Service
- Amazon Kinesis Video Streams
- Amazon Nimble Studio

Migration and Transfer:

- AWS Application Discovery Service
- AWS Application Migration Service
- CloudEndure Migration
- AWS Migration Hub
- AWS Snow Family

Networking and Content Delivery:

- AWS App Mesh
- AWS Cloud Map
- AWS Direct Connect
- AWS Private 5G
- AWS Transit Gateway
- AWS VPN

Quantum Technologies:

- Amazon Braket

Robotics:

- AWS RoboMaker

Satellite:

- AWS Ground Station

Security, Identity, and Compliance:

- AWS Artifact
- AWS Audit Manager
- AWS Certificate Manager
- AWS CloudHSM
- Amazon Detective
- AWS Directory Service
- AWS Firewall Manager
- Amazon GuardDuty
- AWS Network Firewall
- AWS Private CA
- AWS Resource Access Manager (AWS RAM)
- AWS Security Hub
- AWS Shield
- Amazon Verified Permissions

Storage:

- AWS Backup
- Amazon FSx
- Amazon FSx for Lustre
- Amazon FSx for NetApp ONTAP
- Amazon FSx for OpenZFS
- Amazon FSx for Windows File Server
- Amazon S3 Glacier
- AWS Snow Family
- AWS Storage Gateway

Survey

How useful was this exam guide? Let us know by taking our survey.