
THIS MATERIAL IS NOT COMPREHENSIVE

It is only intended as a quick review! This guide is not sufficient on its own for passing the exam; refer to the slides and videos for the finer-grained details you need.

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

Target Candidate Profile

The AWS Certified Generative AI Developer - Professional (AIP-C01) exam validates a candidate's ability to effectively **integrate foundation models (FMs) into applications and business workflows** using AWS technologies.

Recommended experience includes **2 or more years** experience building production-grade applications on AWS or with open-source technologies, general AI/ML, or data engineering. Candidates should also have **1 year of hands-on experience** implementing GenAI solutions. Required foundational knowledge includes AWS compute, storage, networking services, security best practices (identity management), deployment/infrastructure as code (IaC) tools, and monitoring/observability services.

I. Generative AI Fundamentals and Bedrock

Concept	Key Details and AWS Services
Foundation Models (FMs)	FMs are large, pre-trained transformer models. **AWS-supported FMs** include it own **Nova** suite of foundation models, **Jurassic-2 (AI21labs)**, known for multilingual LLMs (Spanish, French, German, Portuguese, Italian, Dutch) for text generation. **Claude (Anthropic)** models are LLMs used for conversations, question answering, and workflow automation. **Stable Diffusion (stability.ai)** handles image, art, logo, and design generation. **Llama (Meta)** is also a supported LLM. **Amazon Titan**

	supports text summarization, Q&A, embeddings, personalization, and search.
Fine-Tuning (Custom Models)	Fine-tuning adapts an existing LLM to a specific use case using additional, potentially proprietary, training data. This eliminates the need for extensive prompt engineering and **saves on tokens** in the long run. **Titan, Cohere, and Meta models** may be fine-tuned via Bedrock. For text models, provide **labeled training pairs of prompts and completions** (e.g., questions and answers). For image models, provide S3 paths linked to image descriptions (prompts). To secure sensitive training data, use a **VPC and PrivateLink**. Fine-tuned models can be used like any other FM.
Retrieval-Augmented Generation (RAG)	RAG functions like an open-book exam for LLMs. It queries an **external database (vector store)** for relevant context (answers) and includes that retrieved text within the prompt for the LLM to process. This method is **faster and cheaper** than fine-tuning for incorporating new or proprietary information, and updating the information only requires updating the database. It leverages **semantic search** via vector stores and helps prevent **hallucinations**. The performance is highly **sensitive to the prompt templates** used and the **relevancy of the information retrieved**.
Knowledge Bases / Vector DB's	Bedrock **Knowledge Bases** facilitate RAG by allowing uploads of documents/structured data from S3, web crawlers, Confluence, Salesforce, or SharePoint. They require an **embedding model** (currently Cohere or Amazon Titan) and a vector store (e.g., **OpenSearch Service**, Aurora, MemoryDB with Valkey, Elasticache with Valkey, MongoDB Atlas, Pinecone, Redis Enterprise Cloud). OpenSearch can function as a vector database. For development, a serverless OpenSearch instance may be used by default.
Optimizing Embeddings & Retrieval	**Chunking** (splitting data prior to storage) is critical, determining how many tokens are represented by each vector. Chunking options in Bedrock include **Hierarchical Chunking** (using smaller child chunks for precision and replacing them with larger parent chunks for context) and **Semantic Chunking** (using an FM to break up content based on semantic content, rather than fixed lengths). **Vector size** (dimensionality) affects cost; optimizing it involves balancing

	dimensionality with retrieval performance. **Metadata** (Document ID, topic, access control) can be stored in the vector DB alongside chunks to improve retrieval relevance scoring and filtering.
Bedrock Guardrails	Provides content filtering capabilities for text FMs. Guardrails filter inputs (prompts) and outputs (responses) based on **word filtering**, **topic filtering**, **profanities**, and **PII removal or masking**. They also perform a **Contextual Grounding Check** to measure how closely the response aligns with retrieved contextual data, which helps prevent hallucinations. Guardrails can be incorporated into agents and knowledge bases.
Token-Level Redaction	Beyond standard Guardrails, token-level redaction involves custom **pre- or post-processing handlers** around inference endpoints. This filters sensitive tokens either before the request hits the model (input filter) or on the output side. Services like **Amazon Comprehend** can be used for Named Entity Recognition (NER) and pattern matching to identify sensitive information.
Prompt Engineering & Flows	A prompt consists of **Instructions, Context, Input data, and an Output indicator**. Techniques include **Few-shot prompting** (providing examples of desired outputs) and **Chain of Thought (CoT)**, which forces the model to "think step by step" for complex reasoning tasks. Bedrock features include **Prompt Management** for storing and versioning reusable prompts that can contain variables. **Bedrock Flows** allows chaining models, prompts, and conditions visually or via JSON to orchestrate complex GenAI logic. Structured JSON output can be enforced by describing the schema in the prompt instructions.

II. Managing Data for Generative AI

Concept	Key Details and AWS Services
Data Structuring	**Raw unstructured text** often loses essential structure like headings and tables. Tools like **Amazon Textract** and **Amazon Comprehend** can help extract structure or covert content into formats like HTML to preserve organization, making it easier for FMs to interpret. **Divider strings** (e.g., <SECTION_BREAK>) can be inserted into documents (e.g., via a Lambda preprocessor or **AWS Glue** ETL pipeline) to improve chunking for vector stores.

Bedrock Data Automation (BDA)	BDA extracts **structured data** from multimodal inputs, including documents, images, videos, and audio. This is used for Intelligent Document Processing (IDP) and preparing data for knowledge bases. BDA uses **Blueprints** to specify exactly what fields the developer wants extracted. Output options include JSON, JSON+files (CSV, markdown), HTML, and CSV. BDA capabilities include **Video Processing** (full summary, chapter summaries, transcripts) and **Audio Processing** (full transcript, speaker labeling, topic breakdown, content moderation).
Amazon Transcribe	Uses Automatic Speech Recognition (ASR) to convert speech to text quickly. It supports **PII Redaction** and **Automatic Language Identification**. Accuracy can be improved using **Custom Vocabularies** (for domain-specific words, brand names, acronyms) and **Custom Language Models** (trained on domain-specific text data to learn context). It also offers ML-powered voice-based **Toxicity Detection** for categories like hate speech and profanity.
Amazon Comprehend	Used for text analysis, NER, and topic modeling. **Custom Classification** organizes documents into user-defined categories. **Named Entity Recognition (NER)** extracts predefined general entities (people, places, organizations). **Custom Entity Recognition** extracts specific terms like policy numbers unique to a business. A Lambda function can integrate Comprehend before data hits Bedrock to **redact PII**, classify, or extract entities.
Vector Store Optimization (OpenSearch)	Dense embedding storage can be resource intensive. Techniques like **Binary vectors** (bit sequences used for distance search, offering 32x compression compared to float32) and **FP16** (storing dimensions as 16-bit values, a form of scalar quantization used by HNSW) reduce size and cost. **OpenSearch Hierarchical Indices** can use a small, fast top-level index to route queries to more detailed, domain-specific indices. The **OpenSearch Neural Plugin** allows OpenSearch to call a Bedrock model to generate embeddings as part of the ingestion or search process.

III. Agentic AI

Concept	Key Details and AWS Services

Bedrock Agents	Agents leverage FMs with tools, planning capabilities, and memory. The **Planning Module** guides the LLM to break down complex requests into sub-questions. **Action Groups** define the tools an agent can use, relying on an **OpenAPI (Swagger) schema** uploaded to S3 to standardize function definitions, input parameters, and outputs, improving tool reliability.
Agent Workflows	Complex tasks may utilize **Multi-agent workflows**, where an **Orchestrator** delegates subtasks to worker LLMs, and a **Synthesizer** combines results. Tasks can be structured as **Chain of Sequence** (sequential processing where output feeds the next step) or use **Parallelization** for concurrent execution (e.g., running multiple guardrails or using voting across multiple model responses). **Model Context Protocol (MCP)** provides a standardized interface (JSON-RPC 2.0 transport over HTTP or stdio) for agent-tool interactions, similar to a universal connector.
Agent Memory	Agents utilize **Short-term memory** (chat history/immediate context via Sessions and Events) and **Long-term memory** (extracted insights, session summaries, user preferences stored as Memory Records/Strategies). **AgentCore Memory** is designed to provide a scalable, serverless solution for storing this information.
Amazon Q Business	A **fully managed Gen-AI assistant for employees** that answers questions and automates tasks based on the company's internal knowledge. It uses **Data Connectors** to crawl data sources like S3, SharePoint, Slack, and Salesforce, and **Plugins** (custom or native for Jira, ServiceNow) for interacting with third-party applications. Authentication via **IAM Identity Center** ensures responses are generated only from documents the user is authorized to access. **Admin controls** (guardrails) allow blocking specific words/topics or restricting responses to internal knowledge only.
Amazon Q Apps	Enables non-coders to rapidly create and share customized Gen AI-powered productivity applications using natural language, leveraging company data and plugins.
Amazon Q Developer	Provides Gen-AI assistance to developers, offering answers based on AWS documentation, suggesting CLI commands, and providing security scans and code generation/completion via IDE extensions (Visual Studio Code, Visual Studio, JetBrains). Project-level rules and guidelines may be stored in the ./amazon/rules directory.

Humans in the Loop (HITL)	Patterns like **Human Augmentation** (AI drafting, human refining) or implementing **Escalation Criteria** (routing uncertain cases based on confidence scores to experts) ensure quality control. User feedback can be collected via a front-end **API Gateway** and stored/indexed in **DynamoDB** to measure model/variant preference and enable continuous improvement.

IV. Operational Efficiency and Optimization

Optimization Area	Key Details and Techniques
Cost & Token Efficiency	Use the **Bedrock CountTokens API** (free) to estimate prompt token count before invoking the model. **CloudWatch** monitors usage via InputTokenCount and outputTokenCount metrics. Techniques include **Context Pruning** (limiting the number of retrieved RAG chunks, filtering irrelevant content via metadata, or summarizing old chat history) and enforcing **Response Size Controls** (using maxTokens or explicit prompt instructions like "respond in 50 words or less"). **Provisioned throughput** for Bedrock may be purchased if consistent performance is required for high workloads; provisioning is associated with a specific model ARN.
Model Selection & Routing	Employ a **Cost/Capability Tradeoff**—a smaller model may suffice if the "smarts" are handled by RAG or tools. Use **Dynamic Routing** (Intelligent Prompt Routing, built into Bedrock) to direct complex queries to larger models and simple queries to smaller, cheaper models. **Bedrock Evaluations** can measure model performance against cost metrics to make informed tradeoffs.
Latency and Caching	**Prompt Caching** (a built-in Bedrock feature) improves latency by caching a static prompt prefix (instructions, system prompt) so only the dynamic content needs to be tokenized on subsequent calls. Cached content is discounted, but writes may be costlier; caching activity is monitored in CloudWatch. Other metrics like **Time to First Token (TTFT)** are tracked by CloudWatch for monitoring latency in streaming responses.
Model Tuning	Tune model parameters based on use case, often evaluated via **Bedrock Evaluations** or **CloudWatch Evidently** (A/B testing). Key parameters include **Temperature** (0 for deterministic, 1 for highly creative/random

	responses), **Top_p** (probability threshold/nucleus sampling), and **Top_k** (sample size of token options).
SageMaker System Optimization	When deploying large models (up to 500GB) via SageMaker AI, adjust **container health check and download timeout quotas** to ensure resources are fully downloaded. Use specialized GPU instance types like ml.p4d.24xlarge for large models or CPU-optimized types like ml.c5.9xlarge for smaller tasks like NER.
System Resiliency	Implementing **Chain of Thought** instructions forces the FM to use reasoning, producing more accurate conclusions for complex tasks. For fault tolerance, although not detailed here, the context suggests using patterns like **Exponential Backoff** for retries and the **Circuit Breaker pattern** (often implemented via Step Functions and DynamoDB) to handle model failures gracefully.

V. Managing Models with SageMaker AI

Concept	Key Details and AWS Services
Model Deployment	Trained models stored in S3 can be deployed to a **Persistent Endpoint** for real-time inference or used for offline prediction via **SageMaker Batch Transform**.
Model Monitoring	**SageMaker Model Monitor** provides alerts (via CloudWatch) on **quality deviations** and visualizes data drift in deployed models (e.g., detecting missing input features or shifting loan approval rates). It detects anomalies and monitors for new features.
Bias and Explainability	Model Monitor integrates with **SageMaker Clarify**, which detects potential bias by measuring imbalances across demographic groups. Bias metrics include **Class Imbalance (CI)** and **Difference in Proportions of Labels (DPL)**. Clarify also aids in **explainability** by identifying which features contribute most significantly to predictions.
Data Labeling	Services like **Amazon Rekognition** (for image recognition) and **Amazon Comprehend** (for text analysis and topic modeling) can automatically generate training labels to prepare data for models.

VI. More Tools for Building AI Applications

Tool/Service	GenAI Relevance and Function

AWS Lambda	A serverless compute service used to encapsulate custom logic for GenAI applications, such as: connecting agents to external tools, performing parameter validation or error handling, invoking FMs on-demand (without provisioning capacity), handling webhooks (e.g., JSON events from API Gateway), and providing custom aggregation logic (like weighted averaging or voting on responses from multiple models).
Amazon AppFlow	A data integration service that facilitates the flow of data between sources (e.g., Amazon S3, Redshift, Snowflake, Marketo) and SaaS applications (e.g., Salesforce, Zendesk). Useful for building ETL pipelines to feed data to or retrieve data from GenAI systems.
AWS CDK	Allows defining cloud infrastructure (IaC) using familiar programming languages (TypeScript, Python, Java). The code is compiled into a **CloudFormation** template, enabling the deployment of infrastructure (like ECS/Fargate containers or Lambda functions) and application code simultaneously.
Amazon Kendra	A fully managed, ML-powered document search service that uses natural language queries. It extracts answers from documents (PDFs, HTML, MS Word) and can learn from user interactions (**Incremental Learning**) and support manual fine-tuning of search results.
API Gateway	Used as the secure front-end for APIs, managing traffic and acting as a proxy for Lambda functions that might interface with GenAI models or collect user feedback.
AWS Transfer Family	Provides fully managed file transfers (SFTP, FTPS, and FTP—within VPC only) directly to Amazon S3 or Amazon EFS. This can be an entry point for ingestion pipelines that feed training or RAG data into the AWS environment.

VII. Governance and QA

Concept	Key Details and AWS Services
Responsible AI	Responsible AI addresses core dimensions including **Fairness, Explainability, Privacy and Security, Safety, Controllability, Veracity and Robustness, Governance, and Transparency**. Tools include **SageMaker Clarify** (for bias detection and explainability), **Bedrock Model Evaluation tools**, and **Amazon Augmented AI (A2I)** for incorporating human review/correction loops.

Evaluation Techniques	**Human Evaluation** is vital due to the non-deterministic nature of GenAI, assessing subjective factors like user experience, creativity, and handling complexity. **Bedrock Evaluation Jobs** measure RAG system performance against benchmarks or LLM judges. Key measured RAG metrics include **Correctness, Completeness, Helpfulness, Logical coherence, and Faithfulness** (how well responses align with retrieved text). Evaluations rely on a prompt dataset, which optionally includes "reference responses" and "reference contexts" (ground truth). Metrics like **ROUGE** measure the overlap of units (words, n-grams) between the generated text and the ground truth for summarization or translation tasks.
Agent Tracing	**Bedrock Agent Tracing** provides visibility into the agent's complex decision-making process. The trace shows the agent's reasoning, detailing what knowledge bases were hit, how action groups were invoked, and any errors encountered. Trace types include **PreProcessing**, **Orchestration**, **PostProcessing**, and **Guardrail** traces.
Observability	**CloudWatch Logs** collect logs from applications, organizing them into log groups and log streams, and supporting encryption via KMS. These logs can be exported to S3, Kinesis, Lambda, or OpenSearch for analysis.

VIII. Security, Identity, and Compliance

The GenAI developer must understand standard AWS security services:

• **IAM** (Identity and Access Management): Manages roles and permissions.

• **AWS KMS** (Key Management Service): Manages encryption keys.

• **Amazon Macie:** A data security and DLP service for discovering and protecting sensitive data.

• **AWS Secrets Manager:** Securely stores and rotates credentials, API keys, and other secrets.

• **Amazon Cognito:** Provides user authentication and authorization for web/mobile apps.

• **AWS WAF:** Protects web applications from common web exploits.

• **Amazon VPC** and **AWS PrivateLink:** PrivateLink ensures private connectivity between VPCs and AWS services (essential for handling sensitive fine-tuning data securely without exposing traffic to the public internet).

IX. Other Services You Should Know

Domain	Key Services and Purpose
Analytics	**Amazon QuickSight** is a serverless business analytics service used for visualizations, paginated reports, ad-hoc analysis, and anomaly detection.
Database	**Neptune Analytics** is an analytics engine built on the Neptune database that allows querying vector databases, using **vectors.topKByEmbedding** to return the top nodes and scores based on explicit vector values. RDS/Aurora and DynamoDB are also general database services referenced.
Management & Governance	**AWS CloudTrail** logs API calls. **AWS Well-Architected Generative AI Lens** guides architecture design across the Generative AI Lifecycle (Scoping, Model Selection, Customization, Integration, Deployment, Continuous Improvement).
Networking & Content Delivery	**Amazon CloudFront** is a Content Delivery Network (CDN) that improves latency and user experience by caching content at hundreds of global edge locations, integrating with AWS Shield/WAF for DDoS protection.
Compute / Containers	**Amazon EMR** allows running big data frameworks like Spark/Hadoop.

X. Exam Preparation

The exam structure includes traditional **Multiple Choice** and **Multiple Response** questions (the latter offering no partial credit). New question types being introduced are **Ordering** (placing 3-5 responses in the correct sequence) and **Matching** (pairing items from two lists). (These types do not appear in the beta exam.) There is no partial credit for the new question types.