CERTIFICATION EXAM PREP · AIF-C01

AWS Certified AI Practitioner

Complete Study Guide · All 5 exam domains

Renata R Silva (Maki) @renatarsilvax is a software engineer, tech content creator and 2x AWS certified (Cloud Practitioner and AI Practitioner). Passionate about communities, she actively shares her software engineering and cloud computing study routine on social media.

Introduction

This guide follows the **official AWS Certified AI Practitioner (AIF-C01) exam guide**, domain by domain, with contrast tables, trigger words, common traps and a quick summary at the end of each domain. The content was built from official AWS pages (full list in the **Sources** section at the end).

Exam domains

Domain	Weight
1. Fundamentals of AI and ML	20%
2. Fundamentals of Generative AI	24%
3. Applications of Foundation Models	28%
4. Guidelines for Responsible AI	14%
5. Security, Compliance, and Governance for AI Solutions	14%

About the exam (official guide)

- 65 questions: **50 scored** and 15 unscored (not identified).
- Question types: multiple choice (1 correct, 3 incorrect) and multiple response (2 or more correct out of 5 or more options).
- Unanswered questions are scored as incorrect; **there is no penalty for guessing**.
- Result: scaled score from 100 to 1,000; **minimum passing score is 700**. Compensatory scoring: you do not need to pass each section, only the exam as a whole.
- Target candidate: up to 6 months of exposure to AI/ML on AWS; uses but does not necessarily build AI/ML solutions.
- Out of scope: developing or coding models, data and feature engineering, hyperparameter tuning, building pipelines, mathematical or statistical analysis of models, implementing security protocols or governance frameworks.

Domain 1: Fundamentals of AI and ML (20%)

Official task statements: 1.1 Explain basic AI concepts and terminologies · 1.2 Identify practical use cases for AI · 1.3 Describe the ML development lifecycle.

1.1 Basic AI concepts and terminologies

The hierarchy: AI, ML, deep learning, generative AI

Think of nested circles: **AI** ⊃ **ML** ⊃ **deep learning** ⊃ **generative AI**. They are layers, not synonyms. According to AWS, **AI** enables machines to perform human-like problem-solving tasks; **machine learning** is a branch of AI that develops algorithms and statistical models to find patterns in historical data; **deep learning** uses neural networks with many layers; **generative AI** creates new content (text, image, video, audio) from prompts.

Term	What it is	Trigger word
AI	Umbrella term: machines performing tasks that normally require human intelligence.	"simulate human intelligence"
ML	Systems that **learn patterns from data** instead of following hand-written rules.	"learns from historical data"
Deep learning	ML with multi-layer **neural networks**; strong with unstructured data (image, text, audio).	"deep neural networks"
Neural network	Model inspired by how the human brain processes information, the basis of modern AI technologies.	"layers of neurons"
Computer vision	Extracts information from images and videos (object detection, faces, content moderation).	"analyze images/videos"
NLP	Natural language processing: interpreting, understanding, and generating human language.	"sentiment, entities, translation"
Model	The trained artifact that receives an input and produces an output (prediction, class, text).	"trained artifact"
Algorithm	The mathematical procedure used *to train* the model.	"learning method"
Training vs. inference	Training = the model learns from data. **Inference** = using the trained model to generate predictions on new data.	"production use phase"
LLM	Large Language Model: a large-scale language model, a type of foundation model for text.	"generates and understands text"
Generative AI	Creates new content. The output is an artifact (paragraph, image, code).	"create, draft, generate"
Agentic AI	Autonomous system that acts independently to achieve predefined goals, with reasoning (LLM), memory, tools, and external data.	"act on its own", "perform multi-step tasks"

Similarities and differences among AI, ML, GenAI, deep learning, and agentic AI

Concept	Main focus	Typical output	Example
AI	Simulate human capabilities	Decisions, actions, content	Voice assistant
ML	Learn patterns from data	Label or number (class, price)	Fraud detection
Deep learning	ML with deep neural networks	Prediction on complex data	Recognizing objects in photos
GenAI	Create new content	Text, image, audio, video, code	Summarizing a document
Agentic AI	Perceive, reason, **act**, and learn with autonomy	Completed tasks (actions)	Agent that queries systems and resolves a ticket

> **Rule of thumb:** if the system produces a **new artifact** (paragraph, image, summary), it is generative AI. If it produces a **label or a number** (fraud/not fraud, predicted price), it is traditional ML, even if there is a neural network underneath. If it **decides and executes actions** in multiple steps with tools, it is agentic AI.

Bias, fairness, and fit

- **Bias** in the data or in the model: the model performs unequally across groups, reproducing or amplifying prejudices present in historical data.
- **Fairness**: equitable treatment across groups; it is the opposite goal of harmful bias.
- Note: "bias" also appears in the statistical sense (high bias = assumptions too rigid = **underfitting**). The context of the question tells you which of the two meanings applies.

Fit	What happens	Symptom	How to fix
Overfitting	The model memorizes details and noise in the training data.	Great on training, **poor on new data** (high variance)	More data, simplify the model, regularization, early stopping, dropout
Underfitting	The model is too simple to capture the pattern (high bias).	**Poor on both training and test**	More complex model, better features, train for longer
Good fit	Captures the pattern and generalizes.	Similar performance on training and test	Goal: balance bias and variance

> **Trap:** excellent training + poor test = overfitting. Poor training + poor test = underfitting. AWS describes overfitting as learning the details and noise of the training data.

Types of inference

Type	When to use	Trigger	Option in Amazon SageMaker AI
Real-time	Interactive workloads with low-latency requirements; persistent endpoint.	"immediate response", "low latency"	Real-time inference (endpoint)
Batch	Large volume of offline data, no urgency, and no persistent endpoint.	"process the entire dataset at once"	Batch transform
Asynchronous	Large payloads (up to 1 GB) and long processing times (up to 1 hour); requests are queued.	"large files", "can take a while"	Asynchronous inference
Serverless	No infrastructure to manage; ideal for idle periods between traffic spikes and workloads that tolerate *cold start*.	"intermittent/unpredictable traffic"	Serverless Inference

Trap: serverless is not "faster": the price is cold start. Asynchronous is not batch: it queues individual requests and responds when done; batch processes an entire dataset.

Types of data in AI models

Data type	Description	Example
Labeled	Each example has a known correct answer (ground truth). Required for supervised learning.	Transactions marked as fraudulent or legitimate
Unlabeled	Raw data without an answer. Used in unsupervised learning.	Millions of loose texts
Structured	Rows and columns with a fixed schema.	Sales table
Unstructured	No fixed schema; the territory of deep learning and GenAI.	Free text, image, audio, video
Tabular	Structured data in a table.	Customer spreadsheet
Time series	Values ordered in time.	Daily sales, sensor readings
Image / text	Inputs for computer vision and NLP.	Product photos, customer reviews

Datasets: training (the model learns), **validation** (hyperparameter tuning and model selection), and **test** (held out, used at the end to estimate real performance). Mixing test with training causes **data leakage**.

Types of learning

Type	How it learns	Example uses	Trigger word
Supervised	With **labeled** data (input → known correct output).	Fraud detection, price prediction, spam classification	"labeled historical examples"
Unsupervised	Finds structure in data **without labels**.	Customer segmentation (clustering), anomaly detection	"no labels", "group similar items"
Semi-supervised	A few labeled data + many unlabeled.	Labeling is expensive, but some examples exist	"few labeled data"
Reinforcement (RL)	An agent acts in an environment and receives **rewards** for correct steps.	Games, robotics, aligning LLMs with human preference (RLHF)	"agent", "reward", "trial and error"
Self-supervised	The model generates its own labels from the data (e.g., predicting the next token).	Pre-training LLMs/FMs on raw text	"no human labels" in FMs

> **Trap:** "unlabeled data" can be unsupervised *or* self-supervised (FM pre-training). If the question mentions agent, reward, and policy, it is reinforcement, never supervised.

1.2 Practical use cases for AI

When AI/ML adds value

- **Assisting human decisions**: recommendations, predictions, triage.
- **Scalability**: handling volumes that would be unfeasible manually.
- **Automation** of repetitive tasks (transcription, image moderation, document classification).
- The pattern is too complex for hand-written rules, representative data exists, and some error is tolerable.

When AI/ML is not appropriate

- When a **specific, deterministic result** is needed, not a prediction (e.g., apply a 10% discount if the order exceeds a fixed amount: a rule is enough).
- When the **cost-benefit analysis** does not pay off (cost of data, training, and operation greater than the gain).
- When there is not enough or representative data.
- When 100% explainable and auditable logic is required (e.g., regulatory calculation).

> **Trap:** in questions with a fixed, simple rule, the right answer is usually "no ML needed".

Choosing the ML technique

Technique	Question it answers	Output	Example
Regression	How much? (continuous numeric value)	Number	House price, next month's demand
Classification	Which category?	Class/label	Fraud or not, spam or not
Clustering	Which items look alike? (no labels)	Groups	Customer segmentation
Anomaly detection	What deviates from the pattern?	Atypical items	Unusual transaction

Real-world applications

- **Computer vision**: object detection, content moderation.
- **NLP**: sentiment analysis, entity extraction, translation.
- **Speech recognition**: call transcription.
- **Recommendation systems**: personalized products and content.
- **Fraud detection** and time series **forecasting**.
- **Knowledge bases**: answers grounded in company documents.
- **Agentic AI**: agents that perform multi-step tasks.

AWS managed AI/ML services (pre-trained, via API)

Service	What it does	Trigger word
Amazon Comprehend	NLP for *understanding*: sentiment, entities, key phrases, PII detection. Not generative.	"analyze sentiment", "extract entities"
Amazon Transcribe	Speech to text.	"transcribe audio/calls"
Amazon Polly	Text to speech.	"natural voice from text"
Amazon Translate	Language translation.	"translate content"
Amazon Lex	Conversational chatbots by voice and text.	"build a chatbot/voice bot"
Amazon Rekognition	Image and video analysis, object and face detection, moderation.	"detect objects in images"
Amazon Textract	Extracts text and data from scanned documents and forms.	"extract data from PDF/form"
Amazon Personalize	Personalized recommendations.	"product recommendation"
Amazon SageMaker AI	Platform to build, train, and deploy your own ML models.	"own labeled data", "custom algorithm"

> **Exam pattern:** if the need is a narrow task (sentiment, translation, transcription) and does not require generating new content, the pre-trained service is usually simpler and cheaper than building a solution with a foundation model.

Traditional ML vs. foundation models (FMs)

Criterion	Traditional ML	Foundation model
Scope	Model specific to one task (e.g., classifying sentiment)	Broad model adaptable to many tasks via prompt
Explainability	Simpler models are easier to explain	Harder to interpret
Regulatory and operational constraints	Preferable when each decision must be explained or the result must be predictable	Requires extra controls (guardrails, human review)
Data	Usually needs its own labeled data	Already pre-trained; adapt with prompt, RAG, or fine-tuning
Infrastructure cost	Lower	Training from scratch is expensive; using it via API (Amazon Bedrock) avoids this

> **Trap:** "regulatory" or "must explain each decision" points to a traditional/simple model. "Multiple tasks", "conversation", "generate content" points to an FM.

1.3 The ML development lifecycle

Components of an ML pipeline

Stage	What happens	AWS tools (examples)
1. Business problem definition	Define the problem and the success metric before touching the data.	-
2. Data collection and preparation	Collect, clean, label, and transform; feature engineering.	SageMaker AI (Data Wrangler, Ground Truth, Feature Store)
3. Model selection and training	Choose the algorithm and train; tune hyperparameters.	Amazon SageMaker AI (training, automatic tuning)
4. Evaluation	Validate with held-out data using technical and business metrics.	SageMaker AI, Amazon Bedrock (model evaluation)
5. Deployment	Make the model available as a service (real-time, batch, asynchronous, serverless).	SageMaker AI endpoints, Amazon Bedrock
6. Monitoring and retraining	Watch performance and drift; retrain when it degrades.	SageMaker AI (model monitoring)

Model sources (including FMs)

- **Open source pre-trained models** (e.g., available through Amazon SageMaker JumpStart).
- **Pre-trained models from providers** accessed through a managed API (Amazon Bedrock).
- **Training a custom model** from scratch with your data (more expensive and requires more expertise).

How to use the model in production

Method	Description	When
Managed API	You call the model through an API without handling infrastructure (e.g., Amazon Bedrock).	Speed, low operational overhead
Self-hosted API	You deploy the model on your own endpoints/infrastructure (e.g., SageMaker AI, with instance choice).	Full control, own model, or model not available in a managed API

AWS services by stage

Need	Service
Access ready-made FMs via API, RAG, agents, guardrails	**Amazon Bedrock**
Build, train, and deploy your own models; host open models	**Amazon SageMaker AI** (includes **SageMaker JumpStart**)
AI assistant for work (create content, research, automate tasks)	**Amazon Quick**
Software development with agentic AI (specs, code generation and verification)	**Kiro**

MLOps fundamentals

MLOps is the set of practices that automates and simplifies ML workflows and deployments, bringing development and operations together across the lifecycle.

Concept	Idea
Experimentation	Test and compare training runs, data, and hyperparameters.
Repeatable processes	Version control and automation to reproduce results.
Scalable systems	Pipelines and infrastructure that grow with demand.
Managing technical debt	Avoid accumulating shortcuts and fragile dependencies in code and ML pipelines.
Production readiness	Testing, CI/CD, and governance before releasing a model.
Model monitoring	Track quality and drift in production.
Retraining	Update the model when performance drops or the data changes.

Benefits cited by AWS: faster time to market, higher productivity through standardized experimentation and better deployment, with centralized versioning and CI/CD.

Model performance metrics

Metric	Formula/idea	Use when
Accuracy	Correct predictions / total predictions	Balanced classes; misleading on imbalanced data
Precision	Of the cases flagged as positive, how many were actually positive	A false positive is costly (e.g., a legitimate email going to spam)
Recall (sensitivity)	Of all real positives, how many the model found	Missing a positive is costly (e.g., fraud, disease)
F1 score	Harmonic mean of precision and recall	A single number that balances both
Confusion matrix	Table of TP, FP, TN, FN	Source of all the metrics above
AUC-ROC	Ability to separate classes across all thresholds; closer to 1 is better	Comparing classifiers

> **Trap:** high accuracy with imbalanced classes can hide a bad model (e.g., 99% of transactions are legitimate). Cost of missing the positive = recall. Cost of a false alarm = precision.

Business metrics

- **Cost per user** and **development costs**.
- **Customer feedback** (satisfaction).
- **ROI** (return on investment): benefit obtained relative to total cost.
- A good technical metric does not guarantee business value: always tie the model to the goal defined in stage 1.

📋 Quick Summary for Domain 1

- AI ⊃ ML ⊃ deep learning ⊃ GenAI: layers, not synonyms. Generates a new artifact = GenAI; label or number = traditional ML; decides and acts with tools = agentic AI.
- Supervised = labeled data. Unsupervised = no labels (clustering, anomalies). Reinforcement = agent + reward. Self-supervised = labels generated from the data itself (FM pre-training).
- Overfitting = good on training, poor on new data. Underfitting = poor on both.
- Training / validation / test kept separate; mixing = data leakage.
- Inference: real-time = low latency; batch = large dataset, no urgency; asynchronous = large/long payload; serverless = intermittent traffic with cold start.
- Regression = number; classification = category; clustering = groups without labels.
- Comprehend (NLP), Transcribe (speech to text), Polly (text to speech), Translate, Lex (chatbot), Rekognition (image), Textract (documents), Personalize (recommendation).
- Pipeline: problem → data → training → evaluation → deployment → monitoring and retraining. MLOps = automation, versioning, monitoring.
- Model sources: pre-trained open models, via managed API, or train a custom one. Production: managed API (Bedrock) or self-hosted (SageMaker AI).
- Precision = fewer false positives. Recall = fewer false negatives. F1 = balance. Business = ROI, cost per user, feedback.

> **Traps:** (1) a fixed, simple rule does not need ML. (2) accuracy misleads on imbalanced data. (3) a narrow pre-trained service usually beats an FM for a single task. (4) explainability/regulation requirements favor a simpler model.

Domain 2: Fundamentals of Generative AI (24%)

Official task statements: 2.1 Explain the basic concepts of generative AI (GenAI) · 2.2 Understand the capabilities and limitations of GenAI for solving business problems · 2.3 Describe AWS infrastructure and technologies for building GenAI applications.

2.1 Basic concepts of generative AI

Key terms

Term	Definition	Trigger
Foundation model (FM)	Large deep learning neural network trained on enormous, broad datasets; performs general tasks and is adapted (prompt, RAG, fine-tuning) instead of trained from scratch.	"general-purpose model"
LLM	Language FM, based on **transformers**, that predicts the next token.	"generates text"
Token	Basic unit the model processes (part of a word). Price, context window, and latency are measured in tokens.	"charged per token"
Context window	Maximum number of tokens the model considers at once. A larger window allows long documents but increases cost and latency.	"very long document"
Chunking	Splitting large documents into smaller pieces (chunks) before generating embeddings, to retrieve only the relevant passages.	"break documents for RAG"
Embeddings / vectors	Numerical representations (vectors) that capture meaning; similar content ends up close together in the vector space.	"semantic search", "similarity"
Vector database	Stores embeddings and performs similarity search (e.g., Amazon OpenSearch Service, Amazon Aurora, Amazon RDS).	"store vectors"
Prompt engineering	Guiding the model toward the desired output by choosing the prompt's format, phrases, and words, without changing the weights.	"improve the response without training"
Multimodal model	Accepts and/or produces more than one modality (text, image, audio, video).	"text + image"
Diffusion model	Generates images by starting from random noise and iteratively removing the noise (e.g., Stable Diffusion).	"text to image"
Transformer	Architecture of modern LLMs; **self-attention** weighs the importance of each token relative to the others.	"LLM architecture"

> **Trap:** chunking and embeddings are preparation steps for RAG and search. The diffusion model generates images; the transformer-based LLM generates text. Embeddings are not the generated text, they are vectors.

Prompting techniques

Technique	Description
Zero-shot	Instruction only, no examples.
Few-shot	A few examples in the prompt to guide format and behavior.
Chain-of-thought (CoT)	Breaks a complex question into intermediate reasoning steps, instead of answering directly.
Prompt templates	Reusable prompts with variables.
Negative prompt	Tells the model what not to include (common in image generation).

> No prompting technique changes the model weights. It is the cheapest and fastest option and requires no training data. Security risk: **prompt injection** (malicious input trying to override the instructions).

Customizing an FM: prompt, RAG, or fine-tuning

Approach	Changes weights?	Solves	Cost/complexity	Data
Prompt engineering	No	Quick behavior adjustment	Lowest	None
RAG	No	Current or private facts; reduces hallucination without retraining	Medium (vector + retrieval)	Your documents (no labels)
Fine-tuning	**Yes**	Specific style, format, task, or jargon	Highest (training)	Labeled example pairs

RAG (Retrieval Augmented Generation): the model consults an authoritative knowledge base outside its training data before responding. Steps: (1) convert external data into vectors in a vector database; (2) turn the question into a vector and retrieve what is relevant; (3) augment the prompt with the retrieved content; (4) keep the external data up to date. Benefits: lower cost than retraining, current information, trust (source citation), and control over the sources. In Amazon Bedrock, the managed feature is **Knowledge Bases**.

In Amazon Bedrock, customization methods include **supervised fine-tuning** (labeled data, adjusts the parameters), **reinforcement fine-tuning** (reward functions instead of labeled pairs), and **distillation** (transfers knowledge from a larger teacher model to a smaller student model, which is faster and cheaper). Continued pre-training is the concept of adapting the model to a domain with unlabeled data.

Exam question	Answer
"Our data changes frequently"	RAG (fine-tuning would require retraining every time)
"Model needs to know internal documents"	RAG
"Adopt a specific tone/format permanently"	Fine-tuning
"Quick adjustment, no training data"	Prompt engineering
"Smaller, cheaper model, keeping the quality of a larger one"	Distillation

Inference parameters

Parameter	What it controls	Low value	High value
Temperature	Shape of the probability distribution for the next token	More deterministic and focused responses	More random and creative responses
Top K	Fixed number of most likely candidates considered	Fewer candidates	More candidates, allows less likely outputs
Top P	Fraction of cumulative probability considered (set size varies per token)	Smaller set	Larger set, more diversity
Response length (max tokens)	Limits the number of tokens in the response	Short response	Long response
Stop sequences	Sequences that make the model stop generating	-	-
Penalties	Penalize repetition, frequency, or types of tokens	-	-

> **Trap:** Top K is a fixed number of candidates; Top P is a cumulative probability threshold. For consistent, factual responses use low temperature. For creativity, high temperature.

GenAI use cases

Use case	Example
Image, video, and audio generation	Create images from text
Summarization	Summarize contracts, meetings, tickets
AI assistants	Questions and answers about company data
Translation	Multilingual content
Code generation	Code suggestions and generation
Customer service agents	Chatbots grounded in documents (RAG)
Search	Semantic search with embeddings
Recommendation engines	Personalized content and products

FM lifecycle

Stage	What happens
1. Data selection	Choose and prepare broad, high-quality data
2. Model selection	Choose architecture/FM based on task, cost, and constraints
3. Pre-training	Train on large volumes of data (self-supervised learning)
4. Fine-tuning	Specialize for a task or domain
5. Evaluation	Measure with automated metrics, human evaluation, and business metrics
6. Deployment	Make the model available (managed API or self-hosting)
7. Feedback	Collect user feedback and monitor to improve the model

Token-based pricing

- Models charge separately for **input** tokens (prompt, context) and **output** tokens (response), generally per million tokens. Price varies by provider/model, Region, and modality.
- **Effect on cost:** long prompts, a lot of retrieved context (RAG), and long responses increase cost. Reducing prompt size and limiting max tokens reduces cost.
- **Effect on performance:** more tokens increase latency. Larger models tend to be more capable, but more expensive and slower.

Context engineering

It is the design of the **content that goes into the FM's context window**: instructions, conversation history, retrieved documents (RAG), tool results, and memory. It goes beyond writing a good prompt: it decides *what* the model sees, in what order, and in what volume, balancing quality, cost (tokens), and latency.

Agentic AI concepts

Concept	Idea
Agent	Autonomous system that acts independently to achieve predefined goals. Cycle: perceive, reason (LLM), act, and learn.
Tool usage	The agent calls APIs, functions, and external systems to execute actions.
Memory	Maintains continuity of the task and context (short and long term).
Workflow orchestration	Coordinates the task's steps, decisions, and tools.
Multi-agent systems	Several specialized agents coordinated, often under a supervisor agent; more scalable and flexible for complex applications.
Multi-agent communication patterns	Ways agents exchange information and delegate tasks (e.g., a supervisor that distributes work to specialist agents).
MCP (Model Context Protocol)	Standard protocol for connecting agents to external systems, tools, and data. In platforms such as Amazon Bedrock AgentCore, the gateway connects agents to MCP servers, APIs, and Lambda functions.

Trap: GenAI "responds"; agentic AI "acts". In questions about connecting agents to external systems in a standardized way, the keyword is MCP.

2.2 GenAI capabilities and limitations for business

Advantages vs. disadvantages

Advantages	Disadvantages
Adaptability to many tasks	**Hallucinations**: plausible but false responses
Responsiveness and speed in generating results	Low **interpretability** (black box)
Conversational capability	**Inaccuracy** and possible bias
Content generation at scale	**Nondeterminism**: the same prompt can produce different outputs
Shorter time to value than building models from scratch	Cost and latency at scale; legal/IP uncertainty about data and outputs

Mitigating hallucination: RAG (ground in sources), lower temperature, human review, and guardrails (such as Guardrails for Amazon Bedrock).

Factors for choosing a model

Factor	What to evaluate
Model type and modalities	Text, image, multimodal, embeddings
Performance requirements and capabilities	Quality on the task, context window size
Constraints and compliance	Data residency, regulation, license, privacy
Cost	Price per token, need for reserved throughput
Latency	Response time required by the application
Model complexity	Larger models = more capable, but more expensive and slower; evaluate a smaller model if it meets the need

Business value and metrics

Metric	What it shows
Cross-domain performance	How well the model performs on different types of tasks
ROI	Return on investment
Efficiency	Time and cost saved to reach the result
Conversion rate	How many visitors take the desired action
Average revenue per user (ARPU)	Revenue generated per user
Accuracy	Correctness of the responses
Customer lifetime value (CLV)	Total expected revenue from a customer

> **Trap:** a model quality metric (accuracy) is not a business metric. To measure impact, use conversion, ARPU, CLV, ROI, and efficiency.

2.3 AWS infrastructure and technologies for GenAI

Services and features

Service	Role	Trigger
Amazon Bedrock	Managed platform to build GenAI apps and agents with FMs from multiple providers via API, without managing infrastructure. Includes Knowledge Bases, Guardrails, agents, customization, and model evaluation. AWS states that it does not store or use your data to train models.	"access FMs via API", "no infrastructure"
Amazon SageMaker AI	Build, train, and deploy your own models, with full control.	"train", "infrastructure control"
Amazon SageMaker JumpStart	Catalog of pre-trained and open models to deploy (including from the Studio interface).	"quickly deploy an open model"
Amazon Quick	AI assistant for work: creates documents and dashboards, researches, automates workflows, and integrates with enterprise applications.	"work assistant"
Kiro	Agentic engineering system: spec-driven development, agents that plan, build, and verify code.	"development with agents"
Strands Agents	Kit for building agents (framework supported by Amazon Bedrock AgentCore).	"build agents in code"
Amazon Bedrock AgentCore	Platform to run agents in production securely: runtime, gateway, identity, observability, and evaluation; any framework and any model.	"agents in production at scale"
Amazon Nova	Amazon's family of FMs.	"Amazon's own models"

Dimension	Amazon Bedrock	Amazon SageMaker AI
Best for	FMs, GenAI apps, speed	Custom models and traditional ML, full control
Infrastructure	Managed, no servers for you to operate	You choose instances and endpoints
Model origin	FMs from providers via API	Your model or JumpStart models
Trigger	"chatbot", "generate text/image", "RAG", "agents"	"own labeled data", "custom algorithm"

> **Trap:** JumpStart lives inside SageMaker AI, but it deploys FMs. If the scenario requires instance control or a model outside Amazon Bedrock, it may be the answer; if it wants a "quick, managed FM", it is Bedrock.

Advantages of using AWS GenAI services

- **Accessibility and lower barrier to entry**: use FMs via API without building infrastructure.
- **Efficiency and cost savings**: no training from scratch; pay for use.
- **Speed to market**.
- Ability to meet business objectives with features such as RAG, agents, and guardrails.

Benefits of AWS infrastructure for GenAI

- **Security**: encryption in transit and at rest; data is not used to train models in Amazon Bedrock.
- **Compliance**: Amazon Bedrock cites adherence to standards such as ISO, SOC, CSA STAR Level 2, GDPR, FedRAMP High, and HIPAA eligibility.
- **Responsibility and safety**: guardrails to filter harmful content and sensitive information.

Cost trade-offs

Option	How it charges	When
On-Demand	Per token, no commitment	Variable or initial usage
Batch inference	Batch processing, lower price than on-demand for selected models (AWS cites a 50% discount)	No urgency
Provisioned Throughput	Hourly, with commitment (1 or 6 months), reserved capacity	High, consistent volume; custom models
Custom models	Training charged by tokens processed (corpus tokens x epochs) + monthly model storage + inference	When prompt and RAG are not enough

- **Responsiveness, availability, and redundancy**: more reserved capacity and cross-Region redundancy increase cost.
- **Performance**: larger models and dedicated throughput cost more.
- **Regional coverage**: models and features vary by Region; price also varies by Region.

> **Trap:** Provisioned Throughput is a time commitment (hourly/1 or 6 months), not "pay per token". On-Demand is the opposite: no commitment and per token.

📋 Quick Summary for Domain 2

- FM = broad pre-trained model, adapted by prompt/RAG/fine-tuning. LLM = text FM based on a transformer (self-attention).
- Token = unit of processing and pricing. Context window = maximum tokens seen at once.
- Embeddings = meaning vectors, stored in a vector database; chunking = splitting documents before embedding.
- Diffusion = noise to image. Multimodal = more than one modality.
- Zero-shot without examples; few-shot with examples; CoT step by step. None changes weights.
- Current/private facts = RAG. Permanent style/format = fine-tuning. Quick adjustment without data = prompt. Smaller, cheaper model = distillation.
- Low temperature = deterministic; high = creative. Top K = fixed number of candidates; Top P = cumulative probability.
- FM lifecycle: data → model → pre-training → fine-tuning → evaluation → deployment → feedback.
- Token pricing: input and output charged separately; long prompts and responses cost more and increase latency.
- Context engineering = deciding what goes into the context window. Agents = tools, memory, orchestration, multi-agent; MCP connects agents to external systems.
- Advantages: adaptability, conversation, content generation. Limits: hallucination, nondeterminism, low interpretability, inaccuracy.
- Business metrics: ROI, conversion, ARPU, CLV, efficiency, cross-domain performance.
- Bedrock = managed FMs via API. SageMaker AI = build/train/deploy. JumpStart = open models. AgentCore = agents in production. Quick = work assistant. Kiro = agentic dev.
- Cost: On-Demand (per token), Batch (lower price), Provisioned Throughput (commitment, reserved capacity).

> **Traps:** (1) RAG changes what the model knows, fine-tuning changes how it behaves. (2) temperature does not fix facts, only variation. (3) Top K (fixed) vs. Top P (cumulative). (4) a technical metric is not a business metric. (5) JumpStart is in SageMaker AI, but it deploys FMs.

Domain 3: Applications of Foundation Models (28%)

Official task statements: 3.1 design considerations for applications that use FMs · 3.2 effective prompt engineering techniques · 3.3 training and fine-tuning process for FMs · 3.4 methods to evaluate FM performance.

> **This is the largest domain of the exam.** The golden rule: AWS rewards the **most managed** and **cheapest option that solves** the problem. The customization ladder, from cheapest to most expensive, is: **prompt engineering → RAG → fine-tuning → continued pre-training → pre-training from scratch**.

3.1 Designing applications with foundation models

FM selection criteria

Criterion	Question to ask	Trigger word
Cost	Price per token (input and output) or per hour (Provisioned Throughput)?	budget, cheap
Modality	Does the model take in/generate text, image, audio, video, or embeddings?	image, multimodal, embeddings
Latency	Near real-time response, or can it be slow?	real time, fast response
Multilingual	Does it support the users' languages?	multiple languages
Model size	Larger models tend to be more capable, but slower and more expensive; smaller ones are faster and cheaper.	lightweight, fast
Model complexity	Does the task require complex reasoning or is it simple (classify, summarize)?	simple vs complex task
Customization	Does the model support fine-tuning, continued pre-training, or distillation in Amazon Bedrock?	adapt to the domain
Input/output size	Does the context window fit the prompt and the expected response?	long document, context
Prompt caching	Is there a long context repeated across requests? The cache reduces latency and input token cost.	repeated context, same document

> **Amazon Bedrock** offers FMs from multiple providers through a **single API**, with no infrastructure to manage, with security and privacy. The **Amazon Nova** models are in scope for the exam. To **compare models** before choosing, use **Amazon Bedrock Model Evaluation** (section 3.4).

Inference parameters

For each token, the model computes a probability distribution and samples from it. The parameters adjust this distribution or limit the response. Default values and ranges depend on the model.

Parameter	Low value	High value
Temperature	More **deterministic** and predictable responses (facts, extraction, code).	More **random and creative** responses (brainstorming, creative writing).
Top K	Considers few candidates (the K most likely tokens).	Considers more candidates, allowing less likely outputs.
Top P	Considers a small slice of the cumulative probability (e.g., 0.7).	Considers a larger slice (e.g., 0.9), more diversity.
Response length (max tokens)	Limits the number of tokens generated (reduces cost and latency).	Allows longer responses.
Stop sequences	Character sequences that stop generation.	-
Penalties	Penalize excessive repetition, frequency, or length.	-

> **Trap:** temperature does **not** change the model's knowledge or fix hallucination; it only changes randomness. For **consistent and factual** responses, use **low** temperature. Input + output size has a limit (context window) and every token is billed.

RAG (Retrieval Augmented Generation)

RAG uses information from data sources to improve the relevance and accuracy of generated responses, **without retraining** the model and without changing its weights. Flow: (1) documents are split into **chunks**; (2) an **embeddings model** converts each chunk into a **vector**; (3) the vectors are stored in a **vector database**; (4) at query time, the question is also turned into a vector and **similarity search** retrieves the closest passages; (5) the passages are added to the prompt and the FM generates the response, which can include **citations** of the sources.

- **Business applications:** support chatbot based on manuals, HR/IT assistant over internal documents, semantic search in a knowledge base, responses with up-to-date and private data.
- **Amazon Bedrock Knowledge Bases:** managed RAG. Connects data sources and handles ingestion, embeddings, and retrieval. It currently offers *Managed Knowledge Base* (AWS manages ingestion, indexing, storage, and retrieval) and *Customer-managed* (you choose and administer the vector store).
- **Embeddings:** numerical representations (vectors) that capture meaning; similar texts end up close together. Amazon Bedrock offers embeddings models (e.g., Amazon Titan Text Embeddings, Cohere Embed). The vector dimension must match the index dimension.

AWS services for storing embeddings (vector databases)

Service	Note
Amazon OpenSearch Service (incl. OpenSearch Serverless)	Vector and hybrid search. The only one (with Serverless) that supports binary vectors in Knowledge Bases.
Amazon Aurora (PostgreSQL) and **Amazon RDS for PostgreSQL**	**pgvector** extension. Ideal if you already use a relational database.
Amazon Neptune (Neptune Analytics)	Graph database; supports **GraphRAG** (relationships between entities).
Amazon S3 Vectors	Vector storage in S3, lower cost, suited to infrequent queries.
Amazon DocumentDB	Also appears in the in-scope services list (document database, MongoDB compatible).
Third parties	Knowledge Bases also support Pinecone, Redis Enterprise Cloud, and MongoDB Atlas.

> **RAG trigger words:** "answer based on company documents", "reduce hallucinations with own data", "up-to-date data without retraining", "cite the source". **Fine-tuning is not the answer** when the data changes frequently.

Cost of customization approaches

Approach	What it does	Cost / effort	When to use
Prompt engineering / in-context learning	Examples and instructions inside the prompt; weights do not change.	**Lowest**; no training.	First attempt, simple tasks.
RAG	Injects external data into the prompt; weights do not change.	Low to medium (vector index + larger token counts).	Private or changing data.
Fine-tuning	Trains with **labeled** data and adjusts weights.	Medium to high (training by tokens × epochs, model storage, Provisioned Throughput).	Specific style, format, or task.
Continued pre-training	Continues pre-training with **unlabeled** domain data.	High.	Adapting to the vocabulary of an entire domain.
Model distillation	Transfers knowledge from a large **teacher** model to a smaller **student**, faster and cheaper.	One-time cost to generate/train; **cheaper inference** afterward.	Maintaining quality with lower cost and latency in production.
Pre-training from scratch	Trains an entire FM.	**Highest** (enormous data and compute).	Rare; only for organizations with extreme resources.

> **Trap:** in Amazon Bedrock, a **custom** model can only be used if you purchase **Provisioned Throughput**. Training is billed by tokens processed (corpus tokens × epochs) and model storage is billed per month.

Bedrock: pricing modes and other application features

Feature	Function	Trigger word
On-demand	Pay for tokens processed, no commitment.	unpredictable traffic, testing
Provisioned Throughput	Dedicated capacity in **Model Units**, billed **per hour**; commitments: no commitment, **1 month**, or **6 months** (longer, bigger discount). Required for a custom model.	high and steady load, custom model
Batch inference	Processes large volumes asynchronously, not in real time.	offline, large volume
Prompt caching	Reuses repeated prompt prefixes to reduce latency and input token cost; it is best-effort (a hit is not guaranteed). On-demand inference only, not batch.	same document across several questions
Guardrails	Configurable safeguards on input and output (details below).	block topic, mask PII

AI agents

An **agent** is an FM able to understand the user's request, **break the task into steps**, collect information through conversation, **call company APIs/systems**, and query knowledge bases to complete the task. In **Agents for Amazon Bedrock**, the main components are **action groups** (actions/APIs the agent can run) and the optional association with a **knowledge base**; you can follow the reasoning through **traces**. Business examples: processing insurance claims, making travel bookings, automating customer service.

> **Scope update:** the official guide lists **Amazon Bedrock AgentCore** and **Strands Agents** (open source SDK for building agents) as in-scope services. The documentation indicates that Amazon Bedrock Agents (now "Agents Classic") is not accepting new sign-ups, and points to agents defined in code with Strands Agents and AgentCore. For the exam, the concept holds: **agent = FM + step-by-step planning + tools/APIs + memory**.

I need to...	Use
Answer based on documents	**Knowledge Bases** (RAG)
Run actions in systems (API, Lambda), multi-step	**Agents** (action groups)
Control/filter what goes into and out of the model	**Guardrails**
Compare models or measure quality	**Model Evaluation**
Reuse and version prompts	**Prompt Management**

Amazon Bedrock Guardrails (controlling what the model says)

Safeguard	What it does
Content filters	Detect harmful content in text/images in these categories: Hate, Insults, Sexual, Violence, Misconduct, and Prompt Attack (prompt attacks). Strength is configurable.
Denied topics	Blocks undesirable topics you define.
Word filters	Block words/phrases (e.g., profanity, competitor names).
Sensitive information filters	Block or **mask PII** (e.g., SSN, address) and accept regex patterns.
Contextual grounding checks	Detect **hallucinations**: a response not grounded in the source or irrelevant to the question (typical in RAG).
Automated Reasoning checks	Validate responses against logical rules.

- They apply to **inputs and outputs**; can be used in FM calls or on their own through the **ApplyGuardrail** API.
- They have a working draft, a test window, and **versions**.
- Question "how to stop the model from talking about X / redact PII / filter toxicity" → **Guardrails**.

3.2 Prompt engineering

Prompt engineering is optimizing the input text of an LLM to get the desired responses. Prompt quality impacts response quality. It does not change the model; it is the **cheapest and fastest** way to improve results.

Prompt concepts and components

Component	Description
Instruction (task)	What the model should do (e.g., "Summarize in one sentence").
Context	Supporting information about the domain or situation, which guides how the input is used.
Input data	The text/data the model acts on.
Examples (demonstrations)	Input/output pairs that calibrate the expected format.
Output indicator / format	How the response should come back (JSON, list, tone, length).
Negative prompt	Says what the model should **not** include/generate (widely used in image generation).

Techniques

Technique	What it is	When to use
Zero-shot	Only the instruction, no examples.	Simple, common tasks.
Single-shot (one-shot)	One example.	Simple format to demonstrate.
Few-shot	Several examples in the prompt (also called *in-context learning*).	Calibrating format/style without training.
Chain-of-thought (CoT)	Asks the model to reason **step by step** before answering (e.g., "think step by step").	Logic problems, math, multiple steps.
Prompt template	Recipe with fixed parts and replaceable fields (e.g., `{{text}}`).	Reusing the same pattern at scale.

> **Best practices:** be **specific and concise**; give clear context; define the output format; use examples when needed; **experiment and iterate** (find out what works by trial); use delimiters/multiple comments or sections to organize; combine with **guardrails** for safety. To reduce hallucinations: optimize the prompt, use RAG, or switch models.

Risks and limitations

Risk	Description
Exposure (prompt leaking)	The model reveals sensitive data or the system prompt/instructions themselves.
Poisoning	Malicious data or prompts introduced to distort the model's behavior.
Hijacking / prompt injection	Malicious instructions embedded in the input hijack the original goal of the prompt.
Jailbreaking	Attempt to bypass the model's safety protections to obtain prohibited content.

- Mitigations: **Guardrails** (includes the Prompt Attack filter), input validation, least privilege for tools, no secrets in the prompt.
- Limitation: the model is **stochastic** (the response can vary) and **does not remember** previous interactions unless the history is sent in the prompt.

Amazon Bedrock Prompt Management

Lets you **create, edit, save, and reuse prompts**. Concepts: **variables** (placeholders filled in at test or run time), prompt **variants** (different message, model, or inference configuration, comparable side by side), **prompt builder** (visual interface in the console), and **versions** (saved snapshots to deploy in a controlled way). It integrates with the application at inference or through a prompt node in **Amazon Bedrock Flows**. It also allows enabling prompt caching.

> **Trap:** "version and manage prompts centrally" → **Prompt Management**. Do not confuse it with **Model Cards** (model documentation) or **Guardrails** (safety).

3.3 Training and fine-tuning FMs

Stages and methods

Stage / method	Data	What it does
Pre-training	Huge volumes of **unlabeled** data (self-supervised learning)	Creates the base FM; the most expensive.
Continued pre-training (continuous)	**Unlabeled** domain data	Continues pre-training so the model learns the vocabulary/knowledge of a domain.
Fine-tuning (supervised)	**Labeled** data (input/output pairs)	Adjusts the weights for a specific task. It is a form of **transfer learning**.
Instruction tuning	Instruction/response pairs	Teaches the model to follow instructions.
Reinforcement fine-tuning (Bedrock)	Prompts + **reward functions** (via AWS Lambda)	Learns from feedback scores instead of labeled pairs.
RLHF	Human feedback (preferences/rankings)	Aligns the model with human preferences (helpfulness, safety); a reward model is trained.
Distillation	Prompts from your use case (responses generated by the teacher)	Transfers knowledge from the large model (teacher) to the smaller one (student).

Fine-tuning vs continued pre-training: labeled and specific task → **fine-tuning**. Lots of raw domain text, no labels → **continued pre-training**. **Transfer learning** = reusing what a pre-trained model has already learned for a new task.

Preparing data for fine-tuning

Aspect	What to ensure
Curation (data curation)	Clean data, without duplicates and noise, of good quality. Quality matters more than volume.
Governance	Privacy, licenses, compliance, provenance, and access control for the data.
Size	Enough examples for the task; too few will not teach, and the model may overfit.
Labeling	Correct and consistent labels (e.g., with **SageMaker Ground Truth**, humans in the loop).
Representativeness	Data covers the real diversity of cases and groups, avoiding bias.
Format	In Bedrock, training data in a file (JSON Lines) in **Amazon S3**.
RLHF	Requires human preference data, used to train the reward model.

Trap: "Garbage in, garbage out": noisy, biased, or unrepresentative data degrade fine-tuning. Fine-tuning with sensitive data requires governance.

Amazon SageMaker AI and related services

Amazon SageMaker AI covers the full ML lifecycle (prepare, build, train, deploy, monitor). **Amazon SageMaker JumpStart** offers pre-trained models and FMs that can be **deployed and fine-tuned** in SageMaker.

I need to...	Use
Managed access to FMs through an API, no infrastructure	**Amazon Bedrock**
Train/build my own model, traditional ML, full control	**Amazon SageMaker AI**
Quickly deploy/fine-tune a pre-trained/open source model on my own instance	**SageMaker JumpStart**
Label data with humans in the loop	**SageMaker Ground Truth**
Prepare/transform data visually	**SageMaker Data Wrangler** (also AWS Glue DataBrew)
Detect bias and explain predictions	**SageMaker Clarify**
Detect drift in production	**SageMaker Model Monitor**
Orchestrate the ML workflow (CI/CD)	**SageMaker Pipelines**
Document the intended use and limitations of the model	**SageMaker Model Cards**
No-code ML for analysts	**SageMaker Canvas**

Trap: Clarify = bias and explainability (before and after training). **Model Monitor** = change over time in production. The documentation indicates that Clarify is no longer open to new customers, but it remains an exam concept. When in doubt between Bedrock and SageMaker, "ready-made foundation model + fast GenAI app" points to **Bedrock**; "train your own algorithm/model" points to **SageMaker**.

AWS assistants and other AI services (for contrast)

Service	Function / trigger word
Amazon Q Business	Generative AI assistant over enterprise data, with responses that respect permissions and include citations; connects to S3, SharePoint, Salesforce, etc. Current documentation states it is no longer open to new customers and cites **Amazon Quick** as an alternative; Quick is in scope for the exam.
Amazon Q Developer	Assistant focused on software development (code suggestions, transformation). Code assistant concept.
Kiro	AI software development tool, listed in scope (Developer Tools category).
Amazon Comprehend	NLP: sentiment, entities, language, PII in text.
Amazon Rekognition	Image and video analysis (objects, faces, moderation).
Amazon Textract	Extracts text, tables, and forms from scanned documents.
Amazon Transcribe	Speech to text.
Amazon Polly	Text to speech.
Amazon Translate	Machine translation.
Amazon Lex	Conversational chatbots by voice and text.
Amazon Personalize	Personalized recommendations.
AWS Transform	In-scope service (workload modernization/transformation), cite by name only.

> These **pre-trained** AI services solve specific tasks without training a model. When the scenario is **open-ended text/image generation**, think **Bedrock**.

3.4 Evaluating FM performance

Evaluation approaches

Approach	Description
Human evaluation (human-in-the-loop)	People assess helpfulness, coherence, correctness, and tone. Gold standard, but slow and expensive. In Bedrock, the team can be employees or industry experts.
Benchmark datasets	Standardized datasets with reference answers to compare models objectively.
Automatic (programmatic) evaluation	Bedrock computes metrics with built-in datasets or your own prompts.
LLM-as-a-judge	A second LLM (judge model) scores the responses and explains each score. Built-in Bedrock metrics: Correctness, Completeness, Faithfulness, Helpfulness, Logical coherence, Relevance, Following instructions, Professional style and tone, Harmfulness, Stereotyping, Refusal. Accepts custom metrics.
RAG evaluation	Evaluates knowledge bases (retrieval and generation) with a question dataset and **ground truth**.

Amazon Bedrock Model Evaluation (Bedrock evaluations): evaluates Bedrock models and knowledge bases, and also models/RAG sources outside Bedrock. Job types: **programmatic**, with **human workers**, with a **judge model**, and **RAG**. Supports FMs, custom and imported models, prompt routers, and models with Provisioned Throughput. Used to **compare models before choosing**.

Metrics

Metric	Measures	Typical use	Limitation
ROUGE (Recall-Oriented Understudy for Gisting Evaluation)	Overlap of n-grams/sequences (focus on **recall**) with a reference text	Automatic **summarization**	Text surface; ignores synonyms and semantics
BLEU (Bilingual Evaluation Understudy)	Overlap of n-grams (focus on **precision**) with the reference	Machine **translation**	Penalizes valid paraphrases
BERTScore	**Semantic similarity** using embeddings	When the words differ but the meaning should be the same	More compute; depends on the embeddings model
Perplexity	How "surprised" the model is by the text; lower is better	Fluency of language models	Does not measure factuality or usefulness
LLM-as-a-judge	Scores from an LLM on criteria (correctness, coherence, relevance...)	Evaluation at scale at lower cost than human	Depends on the quality of the judge

Trap: ROUGE = summarization/recall; **BLEU** = translation/precision; **BERTScore** = meaning (semantics). All compare against a **reference**. None replaces human evaluation for subjective quality.

- **Classification metrics** (when the output is right/wrong): **accuracy**, **precision** (high cost of false positives), **recall** (high cost of false negatives, e.g., fraud), **F1** (balance), and **AUC-ROC**.

Assessing whether the FM meets business objectives

Business metric	Example
Productivity	Time saved per task, volume of tickets resolved per agent.
User engagement	Time of use, returning users, interactions per session.
Task completion rate	Percentage of requests resolved without human intervention.
User satisfaction	CSAT, NPS, positive ratings of the responses.
Cost per interaction	Token and infrastructure cost per conversation handled.
Others	Conversion rate, average revenue per user, customer lifetime value, efficiency (cost and time to reach the target performance). **A/B tests** compare prompt/model versions against a KPI.

Evaluating applications built with FMs (RAG, agents, workflows)

- **RAG:** evaluate **retrieval** (were the right documents brought back? context relevance) and **generation** (is the response faithful to the context? does it answer the question? correctness and completeness). Use Bedrock RAG evaluation and the Guardrails **contextual grounding check** in production.
- **Agents:** evaluate whether they complete the task, whether they choose the right actions/tools, the number of steps, latency, and cost; use **traces** to inspect the reasoning.
- **Workflows** (e.g., Amazon Bedrock Flows): evaluate end to end and per step (prompt, retrieval, tool).
- Combine quality metrics with **latency, error rate, and cost per interaction**.

> **Trap:** model quality (ROUGE, BERTScore) is **not** the same as **business value** (satisfaction, task completion, cost per interaction). The question "does the FM achieve the business objective?" calls for business metrics.

📋 Quick Summary for Domain 3

- **Customization ladder**: prompt engineering → RAG → fine-tuning → continued pre-training → pre-training. From cheapest to most expensive; start with the simplest.
- **FM selection**: cost, modality, latency, multilingual, size, complexity, customization, input/output size, prompt caching.
- **Inference**: low temperature = deterministic; high = creative. Top K/Top P limit candidates. Max tokens, stop sequences, and penalties control length.
- **RAG** = chunks + embeddings + vector DB + similarity search; no retraining. **Knowledge Bases** = managed RAG. Vectors: OpenSearch, Aurora/RDS PostgreSQL (pgvector), Neptune, S3 Vectors.
- **Agents** = FM + steps + APIs (action groups). **Guardrails** = content filters, denied topics, words, PII, grounding, Automated Reasoning.
- **Bedrock pricing**: on-demand (tokens) vs Provisioned Throughput (per hour, 1 or 6 months; required for a custom model). Batch for offline. Prompt caching reduces cost/latency.
- **Prompts**: zero-shot, single-shot, few-shot, chain-of-thought, templates, negative prompts. Risks: exposure, poisoning, hijacking, jailbreaking. **Prompt Management** = variables, variants, versions.
- **Training**: pre-training (unlabeled), continued pre-training (unlabeled domain data), fine-tuning (labeled), RLHF (human feedback), distillation (teacher → student). Data: curation, governance, size, labeling, representativeness.
- **SageMaker** = full ML lifecycle (Ground Truth, Data Wrangler, Clarify, Model Monitor, Pipelines, Model Cards, JumpStart, Canvas). **Bedrock** = managed FMs through an API.
- **Evaluation**: human, benchmark, Bedrock Model Evaluation, LLM-as-a-judge. ROUGE (summarization), BLEU (translation), BERTScore (semantics). Business metrics: task completion, satisfaction, cost per interaction, productivity, engagement.

> **Traps:** (1) using fine-tuning when the problem is changing data (use RAG). (2) thinking temperature fixes hallucination. (3) forgetting Provisioned Throughput for a custom model. (4) mixing up ROUGE and BLEU (summarization vs translation). (5) confusing Clarify (bias/explanation) with Model Monitor (drift). (6) treating quality metrics as proof of business value.

Domain 4: Guidelines for Responsible AI (14%)

Official task statements: 4.1 explain the development of responsible AI systems · 4.2 recognize the importance of transparent and explainable models.

4.1 Development of responsible AI systems

Responsible AI dimensions at AWS

AWS organizes responsible AI into eight core dimensions. The exam may list several and ask you to classify a scenario.

Dimension	Core idea (AWS)	Example scenario
Fairness	Consider the impacts on different groups of stakeholders.	A credit model approves fewer applicants from one demographic group.
Explainability	Understand and evaluate system outputs.	Explaining to the customer why the loan was denied.
Privacy and Security	Appropriately obtain, use, and protect data and models.	Personal data leaking in responses.
Safety	Prevent harmful outputs and misuse.	A chatbot generates toxic or dangerous content.
Controllability	Have mechanisms to monitor and steer the behavior of the system.	A human can intervene, correct, or shut down the system.
Veracity and Robustness	Correct outputs, even with unexpected or adversarial inputs.	Hallucinations; the response breaks with an atypical input.
Governance	Incorporate best practices into the AI supply chain, including providers and deployers.	Policies, auditing, approval before production.
Transparency	Enable stakeholders to make informed choices about their use of the system.	Telling the user they are talking to an AI and documenting limitations.

Characteristics of responsible AI: exam terms

- **Bias**: systematically unequal results for certain groups, almost always originating in the data.
- **Fairness** and **inclusivity**: the system works well for diverse populations, without excluding anyone.
- **Robustness**: reliable performance under varied or adversarial inputs.
- **Safety**: avoiding harm. **Veracity**: the output is truthful (the opposite of hallucination).

Tools to promote responsible AI

Concern	AWS tool	Note
Harmful content, denied topics, PII, hallucination at runtime	**Amazon Bedrock Guardrails**	Filters inputs and responses. See details below.
Bias in data or model; explaining predictions	**Amazon SageMaker Clarify**	Pre-training and post-training bias; feature attribution (Shapley values).
Change in production (quality, bias, attribution)	**Amazon SageMaker Model Monitor**	Monitors data quality, model quality, bias drift, and feature attribution drift.
Human review of uncertain predictions	**Amazon Augmented AI (Amazon A2I)**	Human review workflows for ML predictions, with multiple reviewers.
Labeling and human feedback in the ML lifecycle	**SageMaker Ground Truth**	Applies human feedback throughout the lifecycle.
Compare and evaluate foundation models	**Amazon Bedrock Evaluations** (Model Evaluation)	Automatic evaluation, with an LLM judge or with human reviewers.
Document intended use, risk, and training	**SageMaker Model Cards**	Also **AWS AI Service Cards** (on the AWS side).
Visibility and control over ML models	**SageMaker ML Governance**	SageMaker governance tools, such as Model Cards and the Model Registry.

> **Bedrock Guardrails: what it filters (verified in the documentation). Content filters** (Hate, Insults, Sexual, Violence, Misconduct, and Prompt Attack, with configurable strength); **Denied topics**; **Word filters** (custom words and phrases, e.g., competitor names); **Sensitive information filters** (block or mask PII, with custom regex); **Contextual grounding checks** (detect hallucination in responses not grounded in the source or irrelevant, useful in RAG); **Automated Reasoning checks** (validate responses against logical rules). They evaluate **inputs and outputs**, can be used with the `ApplyGuardrail` API without invoking a model, and you configure the message returned when a filter is violated.

Responsible model selection: sustainability

- Training and hosting large models consumes a lot of energy and compute. Prefer the **smaller model that fits the task** (right-sizing), avoid retraining unnecessarily, and use lightweight customization instead of training from scratch.
- Use efficient hardware (e.g., AWS Trainium and AWS Inferentia) and managed/serverless services to reduce waste.
- The **AWS Well-Architected Generative AI Lens** includes the Sustainability pillar: minimize compute resources in training, customization, hosting, data processing, and storage.

> **Trap:** "train a larger model" is rarely the answer in sustainability or cost questions.

Legal risks of generative AI

Risk	Description	Mitigation
Intellectual property infringement	Training data or outputs that infringe copyrights or licenses.	Licensed or proprietary data, review of model licenses, human review.
Biased outputs	Discrimination against groups, with regulatory and reputational risk.	Diverse data, Clarify, human audits, subgroup analysis.
Loss of customer trust	Public errors, leaks, or inappropriate content.	Guardrails, transparency, testing before release.
End-user risk	Wrong decisions in healthcare, finance, legal.	Human in the loop (A2I), disclaimers, scope limits.
Hallucinations	Plausible but false responses.	RAG, grounding checks, output validation, human review.

Characteristics of good datasets

- **Inclusive and diverse**: represent the affected groups.
- **Curated sources**: known origin, licensed, and high quality.
- **Balanced**: classes and groups without severe underrepresentation.
- **Quality labels**: wrong or biased labels contaminate the model.

Effects of bias and variance

Concept	What it is	Effect
High bias	Model too simple, rigid assumptions.	**Underfitting**: errs on both training and test data.
High variance	Model too sensitive to training details.	**Overfitting**: great on training data, poor on new data.
Bias in the data (fairness sense)	Data that underrepresents or distorts groups.	Inaccuracy and unequal results across demographic groups.

> **Trap:** "bias" has two meanings on the exam: **bias/variance** (model error, underfitting) and **bias/fairness** (discrimination against groups). Read the context.

Detect and monitor bias, trustworthiness, and veracity

- **Analyze label quality** before training.
- **Subgroup analysis**: compare metrics across demographic groups, not just the global average.
- **Human audits** and human evaluation (Bedrock Evaluations, Ground Truth, A2I).
- **Clarify** (bias in data and model) and **Model Monitor** (drift in production).
- **Veracity evaluation**: compare responses to trusted sources, grounding checks, and model evaluation metrics.

4.2 Transparent and explainable models

	Transparent and explainable	Not transparent (black box)
Examples	Linear regression, decision trees; open source models with documented data and license.	Deep networks and large foundation models.
Advantage	You can understand why it decided; easier to audit.	Tends to perform better on complex, unstructured data.
Risk	May perform worse on complex problems.	Hard to explain, audit, and debug.

Transparency tools and practices

- **SageMaker Model Cards**: document in one place the intended and unintended use, risk rating (unknown, low, medium, high), training details, metrics, and evaluation results; they generate immutable versions of edits and can be exported to PDF.
- **AWS AI Service Cards**: AWS documentation on intended use, limitations, responsible AI design decisions, and optimization best practices for its services and models. Difference: Service Cards describe *AWS* services/models; Model Cards document *your* models.
- **Amazon Bedrock Evaluations**: compares models with automatic evaluation, an LLM judge, or human reviewers, to choose based on evidence.
- **Open source models, data, and licensing**: transparency about origin, terms of use, and training data.
- **SageMaker Clarify**: explains predictions by feature importance (Shapley values) and generates reports for risk teams and regulators.

> Availability note: current documentation states that Clarify and Model Monitor are not open to new customers (existing customers continue as normal). For the exam, the concepts and the association "bias/explainability and drift" still apply.

Trade-off between safety, transparency, and performance

- More **interpretable** models tend to sacrifice performance; high-performance models tend to be less explainable. In regulated contexts (credit, healthcare, HR), explainability may weigh more than a few points of accuracy.
- Safety and transparency also pull against each other: exposing too much of how the system works or its data can make attacks easier or leak sensitive information. Evaluate **interpretability and performance** together and choose according to the risk of the use case.

Human-centered design for explainable AI

- **User feedback mechanisms**: thumbs up/down, error reporting, human review (A2I).
- **Transparency in AI decisions**: disclose that it is AI, show sources, explain factors, indicate uncertainty.
- Involve users and stakeholders throughout the lifecycle, not just at the end.

📋 Quick Summary for Domain 4

- 8 AWS dimensions: Fairness, Explainability, Privacy and Security, Safety, Controllability, Veracity and Robustness, Governance, Transparency.
- Bedrock Guardrails = runtime filter (harmful content, topics, words, PII, grounding, Automated Reasoning). Clarify = bias + "why". Model Monitor = drift in production. A2I = human review. Model Cards = documentation.
- Good data: diverse, curated, balanced, with quality labels. Use subgroup analysis and human audits.
- High bias = underfitting; high variance = overfitting.
- Legal risks of GenAI: IP, biased outputs, loss of trust, end-user risk, hallucinations.
- Interpretability versus performance: simple = explainable; deep = powerful and opaque.
- Sustainability: right-sized model, efficient hardware, avoid retraining.

> **Traps:** (1) Clarify, Guardrails, and A2I are not synonyms. (2) Model Cards document your models; AI Service Cards document AWS services. (3) "Bias" can mean fairness bias or bias/variance. (4) A bigger model is not always the answer.

Domain 5: Security, Compliance, and Governance for AI Solutions (14%)

Official task statements: 5.1 explain methods to secure AI systems · 5.2 recognize governance and compliance regulations and requirements for AI systems.

5.1 Securing AI systems

Security services and features applied to AI

Feature	Role in AI solutions
AWS IAM (roles, policies, permissions)	Access control with **least privilege**: who can invoke models, access training data, and change configurations. In Bedrock, you can restrict actions to specific models and resources.
Encryption (AWS KMS)	At rest: training and validation data in S3, custom models, agents, knowledge base ingestion, and evaluations can use KMS keys. In transit: TLS (Bedrock requires TLS 1.2 and recommends 1.3).
Amazon Macie	Discovers and helps protect sensitive data (such as PII) stored in Amazon S3, useful before using that data in training, fine-tuning, or RAG.
AWS PrivateLink and **Amazon VPC**	Interface endpoint (VPC endpoint) to access Bedrock privately, without traversing the public internet; VPC Flow Logs monitor traffic.
AWS Secrets Manager	Stores credentials and API keys used by applications and agents (and third-party vector stores).
Amazon Bedrock Guardrails	Filters inputs and outputs: harmful content, prompt attacks, denied topics, PII, hallucination.
Amazon Bedrock AgentCore Identity	Manages identity, access, and authentication for agents, compatible with existing identity providers (e.g., Amazon Cognito, Okta).
Policy in AgentCore	Deterministic control: defines which tools the agent can use, which actions, and under what conditions, intercepting every tool call via AgentCore Gateway.

> **Trap:** Bedrock is managed, but **IAM, KMS keys, network configuration, and Guardrails configuration remain the customer responsibility**.

Shared responsibility model for AI

	AWS	Customer
Scope	Security *of* the cloud: global infrastructure, hardware, managed service, and model platform.	Security *in* the cloud: IAM, data classification and protection, encryption choices, network, Guardrails configuration, and responsible use.
In Bedrock	Operates the infrastructure and maintains the model deployment accounts; model providers do not have access to customer prompts and responses.	Manages your data, permissions, logs, keys, and how the application uses the model.

Data provenance, source citation, and cataloging

- **Source citation**: indicate where the information came from (e.g., citations in RAG responses) so the user can verify it.
- **Data lineage**: trace the origin and transformations of data up to the model. **Data cataloging**: an organized, searchable inventory of datasets (e.g., AWS Glue Data Catalog).
- **Model Cards**: document training, metrics, intended use, and model risk, supporting audits.

Best practices for secure data engineering

- **Data quality**: completeness, consistency, accuracy, and timeliness before training.
- **Privacy-enhancing technologies**: anonymization, masking, pseudonymization, data minimization.
- **Data access control**: least privilege, environment segregation (e.g., AWS Lake Formation in data lakes).
- **Data integrity**: prevent tampering and data poisoning, with validation and versioning.

Security and privacy considerations

Topic	Threat or requirement	Mitigation
Application security, infrastructure protection	Exposed application and endpoints.	IAM, VPC, PrivateLink, WAF on the front end; shared responsibility model.
Threat detection, vulnerability management	Suspicious activity; known flaws in software and images.	Monitoring and security findings; **Amazon Inspector** for vulnerability scanning.
Prompt injection	Malicious input tries to override the system instructions.	Guardrails Prompt Attack filter, input validation, least privilege for agents.
Encryption at rest and in transit	Interception or unauthorized access.	KMS (at rest) and TLS (in transit).
Data leakage	The model reveals PII or confidential data.	Sensitive information filters, data minimization, Macie, IAM.
Output filtering and validation; toxicity	Harmful, toxic, or out-of-scope responses.	Guardrails, programmatic output validation, human review.
Audit trail and logs of AI interactions	Prove who called what and what was asked and answered.	**CloudTrail** (who and when) + Bedrock **model invocation logging** (content, to S3 and/or CloudWatch Logs).

> **Invocation logging:** it is **disabled by default**. When enabled, it records requests, responses, and metadata of Converse, ConverseStream, InvokeModel, and InvokeModelWithResponseStream calls, delivered to Amazon S3 and/or CloudWatch Logs (same account and Region).

Need	Answer
Block harmful output in real time	Bedrock Guardrails (prevention)
Record what was asked and answered	Model invocation logging (logging)
Record who called which API and when	AWS CloudTrail (auditing)
Metrics, alarms, and operational logs	Amazon CloudWatch (monitoring)

Hallucination detection and grounding techniques

- **RAG grounding**: anchor the response in retrieved documents (Amazon Bedrock Knowledge Bases).
- **Contextual grounding checks** and **Automated Reasoning checks** (Guardrails): detect responses not grounded in the source or that violate logical rules.
- **Output validation**: validate format and facts by rules or by another model (judge).
- **Confidence scoring**: use confidence to route uncertain cases to human review (A2I).

5.2 Governance and compliance

AWS services for governance and compliance

Service	What it is for	Question it answers
AWS Artifact	Self-service portal for AWS compliance reports and agreements.	"Where do I download proof that AWS meets standard X?"
AWS CloudTrail	Records API calls and activity in the account.	Who did what and when?
AWS Config	Assesses and records resource configurations and compliance with rules.	How is the resource configured and what changed?
Amazon Inspector	Automated vulnerability scanning.	Are there known vulnerabilities?
AWS Trusted Advisor	Best practice checks and recommendations (includes security and cost).	Am I following best practices?
Amazon CloudWatch	Metrics, logs, and alarms.	How is the operation doing?
AWS Well-Architected Tool and **Generative AI Lens**	Workload review; the Lens extends the framework to GenAI (operational excellence, security, reliability, performance, cost, sustainability).	Does my GenAI workload follow best practices?

> AWS Audit Manager (continuous auditing) appears in study material, but is not on the official list of in-scope services for this exam.

> **Trap: CloudTrail** = who called which API. **Config** = configuration state and changes. **Artifact** = AWS compliance reports. They are not interchangeable.

Data governance strategies

Topic	What to do
Data lifecycle	Define collection, use, storage, archiving, and disposal (e.g., S3 and S3 Glacier tiers).
Logging	Record access and use of data and of interactions with models.
Data residency	Choose AWS Regions to keep data within the required jurisdiction. Watch for cross-Region inference in the features that offer it.
Monitoring and observation	Track quality, drift, access, and anomalies continuously.
Retention	Keep data and logs for as long as required by policy or law, and no longer.

Governance processes

- Clear **policies** for acceptable use of AI, data, and models; periodic **review cadence** and **review strategies** (including human review).
- **Transparency**: documentation and disclosure standards (Model Cards, AI Service Cards).
- **Team training** in responsible use, security, and compliance.
- **Governance frameworks**, such as the **Generative AI Security Scoping Matrix**.
- Inherited compliance: AWS services hold certifications such as ISO and SOC (reports in AWS Artifact), but configuring your use of them compliantly is the customer responsibility.

Generative AI Security Scoping Matrix

Scope	Description	Ownership
1: Consumer app	Use of a third-party public GenAI service (e.g., chat on a public website).	Lowest
2: Enterprise app	Third-party enterprise application with embedded AI features and enterprise terms.	
3: Pre-trained models	Your own application on a third-party foundation model via API (e.g., chatbot with RAG).	
4: Fine-tuned models	Foundation model fine-tuned with business data.	
5: Self-trained models	Model trained from scratch with your own or acquired data.	Highest

Security disciplines applied across all scopes: **Governance and Compliance**, **Legal and Privacy**, **Risk Management**, **Controls**, and **Resilience**. The higher the scope, the greater your responsibility.

AI governance in SageMaker

- **SageMaker Model Cards**: governance documentation, integrated with the **Model Registry** (model versions and approval workflow).

- **Clarify** generates model governance reports for risk teams and regulators.

📋 Quick Summary for Domain 5

- Standard security stack: IAM (least privilege) + KMS (encryption) + PrivateLink/VPC (private traffic) + CloudTrail (auditing) + Macie (sensitive data in S3) + Guardrails (filters).
- Prevent = Guardrails. Log content = invocation logging (off by default). Log who called = CloudTrail.
- Shared responsibility: AWS protects the cloud; you protect IAM, data, keys, network, and Guardrails configuration, even in Bedrock.
- Prompt injection: Prompt Attack filter, input validation, least privilege for agents (AgentCore Identity and Policy).
- Hallucination: RAG grounding, grounding checks, output validation, confidence, and human review.
- Compliance: Artifact (reports), Config (configurations), CloudTrail (API), Inspector (vulnerabilities), Trusted Advisor (best practices).
- Data governance: lifecycle, residency (Regions), retention, logging, lineage, and catalog.
- Security Scoping Matrix: 5 scopes (consumer, enterprise, pre-trained, fine-tuned, self-trained) and 5 disciplines.

> **Traps:** (1) "AWS takes care of everything in Bedrock" is false. (2) Guardrails prevents; logging records. (3) CloudTrail does not show the content of prompts. (4) Artifact provides AWS reports, it does not audit your configuration (that is Config).

Final Review

Use this chapter the day before the exam: confusable pairs across the 5 domains, traps that cost points, and a self-test to explain out loud.

Contrast table: confusable pairs

Pair	Distinction	Domain
AI vs ML vs Deep Learning vs GenAI	Nested: AI ⊃ ML ⊃ Deep Learning ⊃ GenAI. Output is a label or number: traditional ML. Creates new content: GenAI.	1, 2
Supervised vs unsupervised vs reinforcement	With labels, it predicts. Without labels, it finds structure (clusters). An agent with reward and penalty learns a policy.	1
Overfitting vs underfitting	Overfitting (high variance): great on training data, poor on new data. Underfitting (high bias): poor on both.	1, 4
Precision vs recall	Precision: avoid false positives. Recall: avoid false negatives (e.g., fraud, disease).	1, 3
Prompt engineering vs RAG vs fine-tuning	Prompt: no data, cheapest. RAG: current or private facts, without changing weights. Fine-tuning: style, format, or domain, with labeled data and changed weights.	2, 3
Temperature vs top-p vs top-k	Temperature: overall randomness (0 = deterministic). Top-p: dynamic set by cumulative probability. Top-k: fixed number of candidates.	2
Amazon SageMaker vs Amazon Bedrock	SageMaker: build, train, and deploy models with full control. Bedrock: use foundation models via API, managed, no infrastructure.	3
Bedrock Knowledge Bases vs Agents	Knowledge Bases: managed RAG. Agents: execute multistep tasks and call APIs.	3
Amazon Q Business vs Q Developer	Q Business: assistant over company data. Q Developer: assistant for code.	3
BLEU/ROUGE vs BERTScore	Surface n-gram overlap vs semantic similarity via embeddings.	3
Clarify vs Model Monitor vs Guardrails	Clarify: bias and explainability (data and model). Model Monitor: drift in production. Guardrails: runtime filter of inputs and outputs.	4
A2I vs Ground Truth	A2I: human review of predictions in production. Ground Truth: labeling training data with humans.	3, 4
Guardrails vs Bedrock Evaluations	Guardrails: runtime protection. Evaluations: comparing and evaluating models before choosing.	3, 4
Model Cards vs AI Service Cards	Model Cards (SageMaker): document your models. AI Service Cards: AWS transparency about its services and models.	4
Fairness vs explainability vs transparency	Fairness: impact on groups. Explainability: understand and evaluate outputs. Transparency: stakeholders decide with information.	4
Interpretable model vs black box	Interpretable: explainable, lower performance. Black box: high performance, hard to explain.	4
Prevention vs logging	Guardrails prevents. Invocation logging records content. CloudTrail records who called the API.	5

CloudTrail vs Config vs CloudWatch	CloudTrail: API calls. Config: resource configuration and changes. CloudWatch: metrics, logs, and alarms.	5
AWS Artifact vs AWS Config	Artifact: AWS compliance reports. Config: compliance of your configuration.	5
Macie vs Inspector	Macie: sensitive data in S3. Inspector: software vulnerabilities.	5
KMS vs TLS	KMS: keys for encryption at rest. TLS: encryption in transit.	5
IAM vs PrivateLink	IAM: who can do what. PrivateLink: private network path, no public internet.	5
AWS vs customer responsibility	AWS: security of the cloud. Customer: IAM, data, keys, network, and Guardrails, even in a managed service.	5

10 traps that cost points

1. **ML is not always the answer.** A simple, deterministic rule does not need ML.
2. **RAG vs fine-tuning.** "Data changes frequently" or "use internal documents": RAG. "Permanent tone or format": fine-tuning.
3. **More managed usually wins.** Generic request for a foundation model via API: Bedrock. Own training, custom algorithm: SageMaker.
4. **Clarify, Guardrails, and A2I are not synonyms.** "Responsible AI" is not a single answer.
5. **Guardrails prevents; logging records.** Invocation logging is off by default and blocks nothing.
6. **CloudTrail is not Config.** Who called the API vs how the resource is configured.
7. **"AWS takes care of everything" is almost always wrong.** IAM, data, keys, network, and Guardrails belong to the customer.
8. **Two meanings of bias.** Bias/variance (model error) vs fairness bias. Read the context.
9. **A bigger model is not sustainable.** Right-sizing, efficient hardware, and avoiding retraining are the sustainability and cost answers.
10. **Hallucination is not solved just by changing the model.** Think RAG, grounding checks, output validation, and human in the loop.

> **Trap:** in questions with two plausible answers, choose the one that matches exactly the verb in the question stem: *prevent*, *log*, *explain*, *monitor*, *document*. There is no penalty for guessing: never leave a question blank.

Self-test: explain out loud

1. What is the difference between supervised, unsupervised, and reinforcement learning? Give an example of each.
2. In which scenario is RAG the right choice and in which is fine-tuning better? Why does RAG not change the weights?
3. When should you choose Amazon Bedrock and when Amazon SageMaker? Why does JumpStart blur this choice?

4. What do temperature, top-p, and top-k control? Which temperature value produces more predictable responses?
5. Explain overfitting and underfitting in terms of bias and variance, and how each shows up in the training and test metrics.
6. Say in one sentence what Clarify, Model Monitor, Guardrails, and A2I do.
7. What is the difference between SageMaker Model Cards and AWS AI Service Cards? And between Guardrails and Bedrock Evaluations?
8. Why can "train a larger model" be the wrong answer in sustainability questions? What alternatives exist?
9. Give three responsibilities that remain with the customer when using Bedrock, and say how IAM, KMS, and PrivateLink complement each other.
10. How would you prove to an auditor who called a model, what was asked, and that AWS holds ISO or SOC certification? Name the service for each piece of evidence.

> Reference answers for item 10: CloudTrail (who and when), Bedrock model invocation logging (content), and AWS Artifact (AWS compliance reports).

Sources

Content consulted on **2026-11-22**, from public pages of the **official AWS website** (aws.amazon.com and docs.aws.amazon.com):

- Official AIF-C01 exam guide: https://docs.aws.amazon.com/aws-certification/latest/ai-practitioner-01/ai-practitioner-01.html
- Official exam guide, Domain 1: https://docs.aws.amazon.com/aws-certification/latest/ai-practitioner-01/ai-practitioner-01-domain1.html
- Official exam guide, Domain 2: https://docs.aws.amazon.com/aws-certification/latest/ai-practitioner-01/ai-practitioner-01-domain2.html
- Official exam guide, Domain 3: https://docs.aws.amazon.com/aws-certification/latest/ai-practitioner-01/ai-practitioner-01-domain3.html
- Official exam guide, Domain 4: https://docs.aws.amazon.com/aws-certification/latest/ai-practitioner-01/ai-practitioner-01-domain4.html
- Official exam guide, Domain 5: https://docs.aws.amazon.com/aws-certification/latest/ai-practitioner-01/ai-practitioner-01-domain5.html
- In-scope services: https://docs.aws.amazon.com/aws-certification/latest/ai-practitioner-01/aif-01-in-scope-services.html
- What is Artificial Intelligence?: https://aws.amazon.com/what-is/artificial-intelligence/
- What is Machine Learning?: https://aws.amazon.com/what-is/machine-learning/
- Deploy models for inference (SageMaker AI): https://docs.aws.amazon.com/sagemaker/latest/dg/deploy-model.html
- What is Prompt Engineering?: https://aws.amazon.com/what-is/prompt-engineering/
- What are Foundation Models?: https://aws.amazon.com/what-is/foundation-models/
- What is RAG?: https://aws.amazon.com/what-is/retrieval-augmented-generation/
- Inference parameters (Amazon Bedrock): https://docs.aws.amazon.com/bedrock/latest/userguide/inference-parameters.html
- What is MLOps?: https://aws.amazon.com/what-is/mlops/
- Amazon Bedrock pricing: https://aws.amazon.com/bedrock/pricing/
- Amazon Bedrock: https://aws.amazon.com/bedrock/
- Customize your model (Amazon Bedrock): https://docs.aws.amazon.com/bedrock/latest/userguide/custom-models.html
- What is Agentic AI?: https://aws.amazon.com/what-is/agentic-ai/
- Amazon Bedrock AgentCore: https://aws.amazon.com/bedrock/agentcore/
- Amazon Quick: https://aws.amazon.com/quick/
- Amazon Bedrock Guardrails: https://docs.aws.amazon.com/bedrock/latest/userguide/guardrails.html
- Amazon Bedrock evaluations: https://docs.aws.amazon.com/bedrock/latest/userguide/evaluation.html
- Amazon Bedrock Prompt management: https://docs.aws.amazon.com/bedrock/latest/userguide/prompt-management.html
- Amazon Bedrock Knowledge Bases: https://docs.aws.amazon.com/bedrock/latest/userguide/knowledge-base.html

- Amazon Bedrock Provisioned Throughput: https://docs.aws.amazon.com/bedrock/latest/userguide/prov-throughput.html
- Amazon Bedrock prompt caching: https://docs.aws.amazon.com/bedrock/latest/userguide/prompt-caching.html
- Amazon Bedrock Agents: https://docs.aws.amazon.com/bedrock/latest/userguide/agents.html
- Amazon Bedrock prompt engineering concepts: https://docs.aws.amazon.com/bedrock/latest/userguide/prompt-engineering-guidelines.html
- Amazon Q Business: What is: https://docs.aws.amazon.com/amazonq/latest/qbusiness-ug/what-is.html
- Amazon Bedrock Knowledge Bases vector stores: https://docs.aws.amazon.com/bedrock/latest/userguide/knowledge-base-setup.html
- Amazon Bedrock evaluation metrics (judge): https://docs.aws.amazon.com/bedrock/latest/userguide/model-evaluation-metrics.html
- SageMaker Clarify: https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-configure-processing-jobs.html
- SageMaker JumpStart foundation models: https://docs.aws.amazon.com/sagemaker/latest/dg/jumpstart-foundation-models-use.html
- Responsible AI on AWS: https://aws.amazon.com/ai/responsible-ai/
- AWS AI Service Cards (resources): https://aws.amazon.com/ai/responsible-ai/resources/
- Amazon SageMaker Model Cards: https://docs.aws.amazon.com/sagemaker/latest/dg/model-cards.html
- SageMaker Model Monitor: https://docs.aws.amazon.com/sagemaker/latest/dg/model-monitor.html
- Amazon Augmented AI: https://aws.amazon.com/augmented-ai/
- Bedrock model invocation logging: https://docs.aws.amazon.com/bedrock/latest/userguide/model-invocation-logging.html
- Bedrock data protection: https://docs.aws.amazon.com/bedrock/latest/userguide/data-protection.html
- Bedrock data encryption: https://docs.aws.amazon.com/bedrock/latest/userguide/data-encryption.html
- Bedrock VPC and PrivateLink: https://docs.aws.amazon.com/bedrock/latest/userguide/usingVPC.html
- Generative AI Security Scoping Matrix: https://aws.amazon.com/ai/security/generative-ai-scoping-matrix/
- Well-Architected Generative AI Lens: https://docs.aws.amazon.com/wellarchitected/latest/generative-ai-lens/generative-ai-lens.html
- Amazon Bedrock AgentCore overview: https://docs.aws.amazon.com/bedrock-agentcore/latest/devguide/what-is-bedrock-agentcore.html

> **Transparency: AWS Skill Builder** requires a login and was not accessed directly, so please cross-check the content there. Some short descriptions were written from standard AWS documentation. Numbers and service names change over time: confirm them on the official pages before the exam.