

AWS Certified AI Practitioner

AIF-C01

FREE PRACTICE EXAM

Ready to test
your readiness?

Take the free AIF-C01 exam →

Learn it. Recognize it. Compare it. Pick the right answer. Everything the exam actually tests, squeezed into pages you can review in an afternoon.

Aligned to Exam Guide v1.1 (Apr 30, 2026)

Concept recognition + service selection

Not an exam dump

The exam at a glance

Questions	65 total: 50 scored + 15 unscored (you can't tell which)
Passing score	700 on a 100 to 1,000 scale
Scoring	Compensatory. You pass the exam overall, not each domain
Guessing	No penalty. Never leave a question blank
Who it's for	People who use AI on AWS (up to 6 months in), not people who build models
Formats	Multiple choice, multiple response, ordering, matching

Where the points live

20	24	28	14	14
D1 Fundamentals of AI & ML	20%	p. 2-4		
D2 Fundamentals of GenAI	24%	p. 5-8		
D3 Applications of FMs	28%	p. 9-15		
D4 Responsible AI	14%	p. 16-17		
D5 Security & governance	14%	p. 18-20		
Command Center		p. 21-24		

How to use this sheet

- **First pass (3 to 4 hours).** Read pages 2 to 20 once, in order. Focus on the blue clue boxes and the comparison tables.
- **Second pass.** Skim only the clue boxes and amber traps. If one surprises you, circle it.
- **Last 48 hours.** Pages 21 to 24, plus any domain where your practice scores sit under 75%.
- **Practice exams** tell you what you don't know. This sheet fixes it.

The four callouts you'll see

CLUE Wording in a question that points to one answer.

TRAP Two things the exam loves to blur together.

COACH TIP How I'd reason through it in the exam room.

FORMAT What the question type is asking you to do.

How the four question types actually play out

MULTIPLE CHOICE One right answer, three distractors. Kill the two obviously wrong ones first, then compare what's left against the *exact* requirement.

MULTIPLE RESPONSE Two or more right answers out of five or more. You need all of them. Treat each option as true or false on its own.

ORDERING 3 to 5 steps in sequence. Think workflows: RAG pipeline, ML pipeline, FM lifecycle. Know the order cold.

MATCHING Pair 3 to 7 prompts with answers. Usually service-to-use-case or term-to-definition. One slip costs the whole item.

What changed in Exam Guide v1.1

- **New topics:** token-based pricing, context engineering, agentic AI (MCP, memory, tools, multi-agent), Bedrock Prompt Management, traditional ML vs. foundation models, business alignment metrics, hallucination detection and grounding.
- **New in-scope services:** Amazon Aurora, Bedrock AgentCore, Kiro, Strands Agents, Amazon Q, SageMaker JumpStart, AWS Transform. Amazon Quick and Nova are also on the list.
- **Gone or demoted:** AUC as a named metric, MemoryDB, and the old SageMaker pipeline examples (Data Wrangler, Feature Store, Model Monitor).

Coach's note before you start

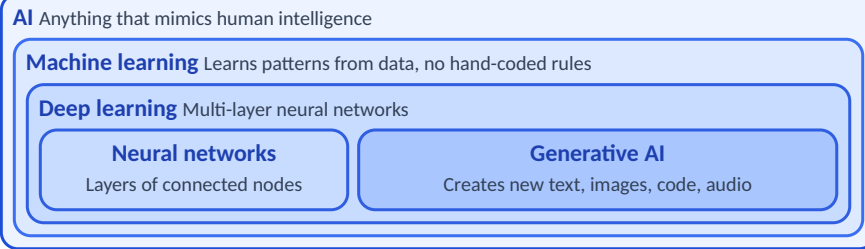
AIF-C01 tests **recognition**, not building. Nobody's asking you to write a prompt template from scratch or configure a vector index. They're asking, "Here's a business problem. What's the right concept, and what's the right AWS service?"

If a question starts to feel like a deep technical configuration task, back up. The answer is almost always the simplest managed option that fits the requirement.

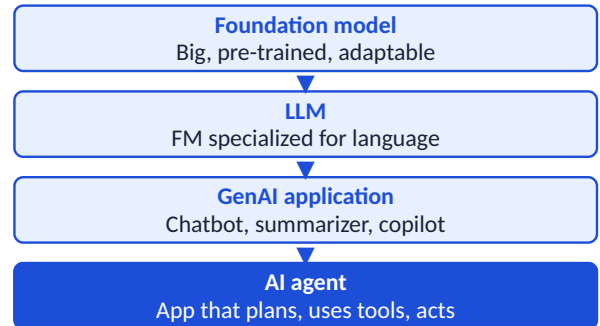
Scope source: AWS Certified AI Practitioner (AIF-C01) Exam Guide v1.1. Service descriptions checked against current AWS documentation.

★ **MUST MEMORIZE** Category → classification. Number → regression. Groups → clustering. Real-time = someone's waiting. Batch = nobody is.

VISUAL 1 The family tree



VISUAL 1B The GenAI stack



AI vs. ML vs. deep learning vs. GenAI vs. agentic AI

Term	Plain English	Example	Exam clue
AI	The umbrella. Machines doing tasks that normally need human judgment.	Rules-based fraud screen, voice assistant	"Broadest" term
ML	A subset of AI. The system learns from data instead of following hand-written rules.	Predicting customer churn	"Learns from data"
Deep learning	ML with many-layered neural networks. Great with images, audio, and text.	Image classification	"Neural network," "layers"
GenAI	Deep learning models that <i>create</i> new content instead of just labeling or predicting.	Summarizing a contract	"Generate," "create," "draft"
Agentic AI	Systems that chase a goal on their own: they plan, call tools, take actions, and adapt.	Agent that files an insurance claim end to end	"Autonomous," "multi-step," "takes action"

Terms you can't be fuzzy on

Model	The trained artifact you actually call.
Algorithm	The recipe used to learn. Algorithm + data = model.
NLP	Understanding and producing human language.
Computer vision	Understanding images and video.
LLM	FM trained on huge text datasets. Predicts the next token.
Foundation model	Broadly pre-trained, reusable across many tasks.

Training vs. inference

	Training	Inference
What happens	Model learns from data	Model answers new input
Frequency	Occasional	Every request
Compute	Heavy, upfront	Light each time, adds up
Output	A model	A prediction or generation
GenAI billing	Rarely your problem	Per token

TRAP "Using the model" is inference, even if the question never uses the word.

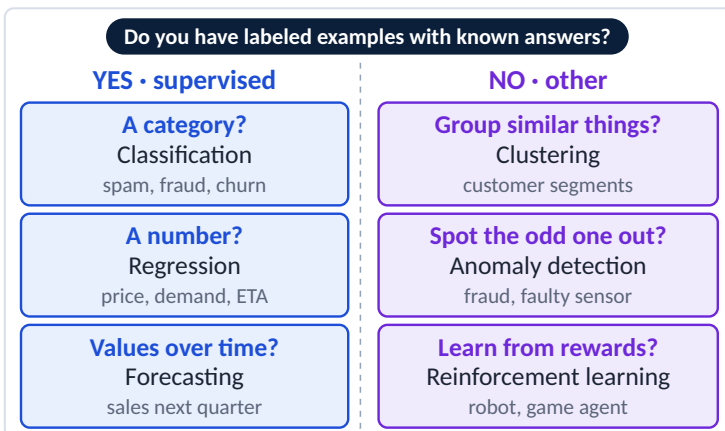
Inference types: how fast, how big, how often?

Type	How it behaves	Pick it when	Exam clue
Real-time	Always-on endpoint, answers immediately	Users are waiting: chat, checkout fraud check	"low latency," "interactive"
Batch	Runs over a big dataset on a schedule, results saved	No one's waiting: nightly scoring, bulk document processing	"large dataset," "no immediate response"
Asynchronous	Request goes in a queue, result comes back later	Big payloads or long processing, but you still want it soon	"large payload," "long processing time"
Serverless	No capacity to manage, scales to zero between calls	Spiky or unpredictable traffic that can tolerate a cold start	"intermittent traffic," "don't manage infrastructure"

CLUE Real-time = someone is waiting. Batch = nobody is. Async = big or slow, but still wanted soon. Serverless = unpredictable traffic, no servers to babysit.

VISUAL 2 Supervised vs. unsupervised vs. reinforcement learning

Type	Data	Goal	Typical tasks	Exam clue
Supervised	Labeled	Predict a known kind of answer	Classification, regression	"labeled," "historical outcomes"
Unsupervised	Unlabeled	Find structure nobody told it about	Clustering, anomaly detection	"segments," "no labels"
Reinforcement	No dataset; feedback from an environment	Learn which actions earn the most reward	Robotics, games, dynamic decisions	"reward," "trial and error," "agent"

VISUAL 3 Which ML technique? Walk the tree

Classification vs. regression vs. clustering

	Output	Labels?	Ask yourself
Classification	A category	Yes	"Which bucket?"
Regression	A number	Yes	"How much? How many?"
Clustering	Groups	No	"What naturally goes together?"

TRAP Predicting yes/no or fraud/not fraud is **classification**, even though the answer feels like a number (1 or 0). Regression only wins when the answer is a continuous quantity.

COACH TIP Look at the *answer* the business wants, not the data. Category, number, or group. That decides it in about five seconds.

Data types: what you're feeding the model

Kind	What it looks like
Structured	Rows and columns with a fixed schema. Tabular data, SQL tables, spreadsheets. Time-series data (readings stamped over time) lives here too.
Semi-structured	Some organization, no rigid table. JSON, XML, logs, email headers.
Unstructured	No predefined shape. Text, images, audio, video , PDFs. Most GenAI work starts here.
Labeled	Each example carries the right answer. Required for supervised learning.
Unlabeled	Raw examples, no answers. Unsupervised learning and FM pre-training.

Traditional ML or a foundation model?

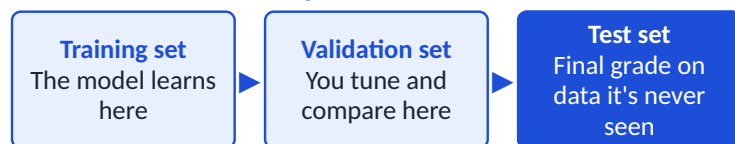
Lean traditional ML when...	Lean FM when...
Regulators or auditors need explainable decisions	You need to generate or transform language, images, or code
It's a narrow prediction on structured data	One model has to handle many tasks
Latency, cost, or operating limits are tight	You have little labeled data
You need the same output every time	You want a working prototype fast

Business problem → ML approach → AWS service you'd name

The business wants to...	Approach	AWS service the exam usually pairs with it
Recommend products based on behavior	Recommendation	Amazon Personalize
Pull text and fields out of scanned forms and invoices	Document analysis (OCR+)	Amazon Textract
Find objects, faces, or unsafe content in images and video	Computer vision	Amazon Rekognition
Detect sentiment, entities, or topics in text	NLP	Amazon Comprehend
Spot fraud or forecast demand (use-case recognition only)	Anomaly detection / forecasting	Fraud Detector and Forecast exist; don't study them deeply
Write, summarize, or chat with company data	Generative AI	Amazon Bedrock

WHEN AI/ML IS NOT THE ANSWER The outcome has to be exact, not a prediction (tax math, a fixed policy rule). The cost of building it beats the benefit. There's simply not enough usable data. If a plain rules engine solves it, the exam wants that.

Three datasets, three jobs



CLUE "Evaluate on data the model hasn't seen" means the **test** set. Scoring a model on its own training data tells you almost nothing.

TRAP: TWO MEANINGS OF "BIAS" **Bias (ML sense)** = error from a model that's too simple, the underfitting side of the bias/variance tradeoff. **Bias (fairness sense)** = unfair outcomes for certain groups, which is Domain 4. Read the context before you answer.

Fixing a model: what the exam expects

Problem	Move
Overfitting	More and more varied data, simpler model, stop training sooner
Underfitting	Better features, more capable model, train longer
Biased outputs	Balanced, representative data; subgroup testing

CLUE: PRECISION OR RECALL? Missing a real case is costly (fraud, disease, safety) → **recall**.

False alarms are costly (blocking good customers, spam folder eating real mail) → **precision**.

Need a balance, or classes are lopsided → **F1**.

Overfitting vs. underfitting

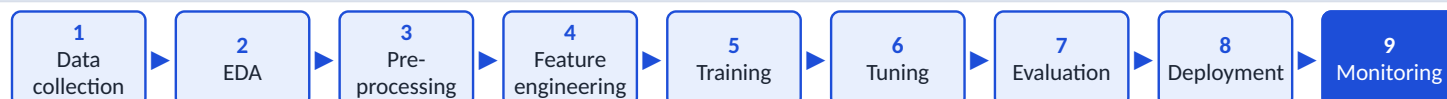
	Training	New data	Culprit
Underfit	Poor	Poor	Too simple. High bias
Good fit	Good	Good	Balanced
Overfit	Great	Poor	Memorized noise. High variance

Metrics: say it in plain English

Metric	Plain English
Accuracy	Share of all predictions that were right
Precision	Of the ones flagged positive, how many truly were
Recall	Of the real positives, how many did you catch
F1	One score balancing precision and recall

TRAP Accuracy lies on lopsided data. If 99% of transactions are legit, a model that says "not fraud" every time is 99% accurate and completely useless.

VISUAL The AI/ML pipeline: know the order



Monitoring feeds back into retraining. EDA = exploring the data for patterns and problems before you model it. This is a recognition topic: know what each stage does and the order. You won't be asked to perform any of it.

Where do foundation models come from? How do you run them?

Source	Notes
Open-source pre-trained	Free to start from, you host and manage it
Provider's model via API	Fastest. This is Amazon Bedrock
Train your own	Most control, most cost and effort
Run in production	Notes
Managed API service	AWS runs it for you (Bedrock)
Self-hosted API	You deploy and operate the endpoint

Which AWS service fits which stage? (1.3.4)

Service	Role in the pipeline
Amazon Bedrock	Use and customize FMs through an API
SageMaker AI	Build, train, deploy your own models
Amazon Quick	Business users get answers, BI, automation
Kiro	Spec-driven agentic IDE and CLI for developers
Amazon Q	Named in this objective. Recognition only (see p. 8)

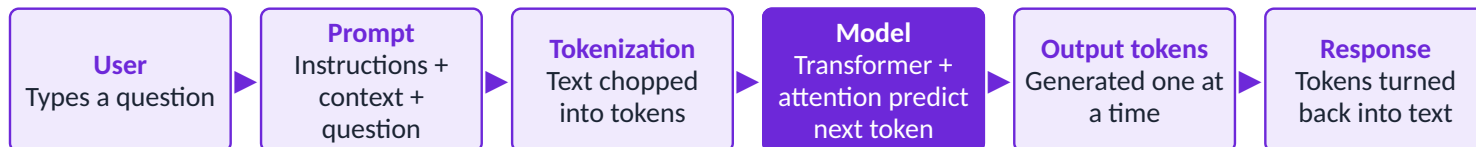
MLOps in one breath, plus business metrics

MLOps means running ML like real software: **experiment** and track what you try, make steps **repeatable**, build **scalable** systems, keep **technical debt** down, reach **production readiness**, **monitor** models, and **retrain** when performance drifts.

Business metrics answer "was it worth it?": cost per user, development cost, customer feedback, ROI. A model can score beautifully and still fail the business case. The exam likes that contrast.

★ **MUST MEMORIZE** An LLM predicts the next token. It doesn't look facts up, which is exactly why it can hallucinate.

VISUAL 4 One request, start to finish



CLUE The model doesn't "look up" an answer. It predicts the most likely next token, over and over. That one idea explains hallucinations, randomness, and why the context window matters.

The vocabulary, one line each

Term	What it means	Remember this
Token	A small chunk of text (a word or piece of one) the model reads and writes	What you pay for and what limits length
Tokenization	Splitting text into tokens before the model sees it	Step one inside the model
Context window	The maximum tokens the model can handle at once: prompt, history, and response together	Too much input gets cut off or won't fit
Embedding	A list of numbers (a vector) that captures the <i>meaning</i> of text, an image, or other data	Similar meaning = nearby vectors
Vector database	Storage built to find the nearest vectors fast	Powers semantic search and RAG
Chunking	Cutting long documents into smaller pieces before embedding	Step two of RAG
Transformer	The architecture behind modern LLMs. Uses attention to weigh which words matter to each other	"Attention" = context awareness
Multimodal model	Handles more than one kind of input or output: text, images, audio, video	"Describe this photo" tasks
Diffusion model	Creates images by starting from noise and removing it step by step	The image-generation workhorse
Prompt engineering	Shaping the input to get a better output without changing the model	Cheapest tool you have (p. 10)

Embeddings vs. tokens

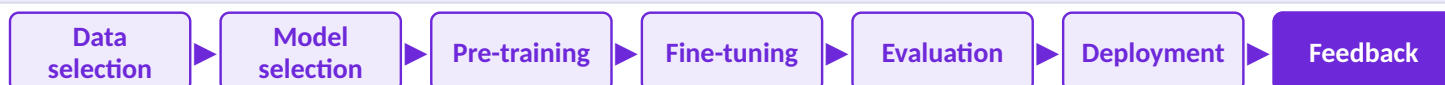
	Token	Embedding
It is	A piece of text	A vector of numbers
Captures	Just the text chunk	Meaning
Used for	Model input/output, billing, limits	Similarity search, RAG
Clue	"context limit," "cost"	"semantic," "similar"

Foundation model vs. fine-tuned model

	Foundation	Fine-tuned
Trained on	Huge, broad data	FM + your smaller labeled set
Good at	Many general tasks	Your domain, tone, format
Effort	Use as-is	Extra training and cost
Clue	"general-purpose"	"organization-specific behavior"

TRAP Tokens get **counted**. Embeddings get **compared**.

VISUAL The foundation model lifecycle



An ordering question likes this one. Feedback loops back into data and fine-tuning. Pre-training on broad data happens *before* fine-tuning on specific data.

Use case → what GenAI is doing underneath

Use case	Capability	Real example
Summarization	Text generation	Boil a 40-page contract down to a page
Translation	Language generation	Localizing product descriptions
Code generation	Code generation	Drafting functions, tests, fixes
Chatbot	Conversational AI	HR policy Q&A bot
Image creation	Image generation (diffusion)	Marketing mockups
Video / audio generation	Video / audio generation	Voiceover, short clips
Search	Retrieval + generation (RAG)	Ask questions over internal docs
Recommendations	Personalization	Tailored product suggestions
Customer service	AI assistant or agent	Resolves a refund without a human

WHY BUSINESSES LIKE GENAI Adaptable across tasks, quick to respond, conversational, and good at producing first drafts at scale.

WHAT THE EXAM ALSO WANTS NAMED Hallucinations, bad interpretability, inaccuracy, and nondeterminism are the standard disadvantages.

CHEAT CHART GenAI limitations: problem → why → fix

Problem	Why it happens	What you do about it
Hallucination	The model predicts plausible text, not verified truth	Ground it with RAG, require citations, human review
Bias	Skewed or unrepresentative training data	Better data, evaluation across groups, guardrails
Toxicity	Harmful content exists in training data	Guardrails and content filters
Incorrect information	Training data is old or wrong	Connect to current sources (RAG), verify outputs
No factual grounding	It knows patterns, not your company's facts	Retrieval from approved documents
Context limits	Context window caps how much it can consider	Trim the prompt, chunk, retrieve only what's relevant
Non-deterministic	Sampling adds randomness to each token	Lower the temperature (p. 11)
Data privacy	Sensitive text can end up in prompts or outputs	Access controls, masking PII, encryption, guardrails
IP concerns	Output may echo copyrighted or licensed material	Data and model governance, license review
Cost	You pay per token; big models and long prompts add up	Smaller model, shorter prompts, caching
Latency	Bigger models and longer outputs take longer	Smaller model, cap output length

CLUE "Confident but wrong answer" = hallucination. "Same prompt, different answers" = non-determinism. "Offensive output" = toxicity. Three different problems, three different fixes.

TRAP Hallucination is not bias. Bias is *unfair* output tied to who or what the data under-represented. Hallucination is *made-up* output. Fixing one doesn't fix the other.

Which GenAI task? A quick sort

Create something new

Draft, summarize, translate, generate code or images

Answer from your data

RAG over documents, grounded Q&A, search

Take action

Agents that call tools, update systems, finish workflows

Token-based pricing: what drives the bill

- You're charged for **input tokens** (your prompt, history, retrieved text) and **output tokens** (the answer).
- Bigger models cost more per token than smaller ones.
- Longer prompts and longer answers cost more and respond slower.
- Steady, heavy traffic can justify **reserved capacity** (Provisioned Throughput) instead of pay-as-you-go.

WAYS TO CUT COST Shorter prompts, a smaller model that still meets quality, a max output length, prompt caching for repeated context, retrieving only the relevant chunks.

Prompt engineering vs. context engineering

Prompt engineering is *how you ask*. Wording, examples, format.

Context engineering is *what the model gets to see*. You decide which instructions, retrieved documents, conversation history, tool results, and memory go into the context window, and how much.



The goal is the right information, in the right amount. More isn't better.

Agentic AI: the concepts you need to recognize (2.1.6)

Concept	What it is	Clue in a question
AI agent	An FM wrapped with instructions, tools, and memory that works toward a goal	<i>"acts on its own," "completes tasks"</i>
Tools / actions	What the agent can do: call an API, query a database, send an email, run code	<i>"call external systems"</i>
Planning	The agent breaks a goal into steps and decides what to do next	<i>"multi-step," "decides how"</i>
Memory	Short-term: the current conversation. Long-term: facts kept across sessions	<i>"remembers," "personalized over time"</i>
Orchestration	Coordinating the steps, tools, and models in a workflow	<i>"coordinates," "workflow"</i>
Multi-agent	Several agents with different jobs. Often a supervisor hands work to specialists	<i>"specialized agents collaborate"</i>
MCP (Model Context Protocol)	An open standard for connecting agents to tools and data sources, so you don't hand-build every integration	<i>"standard way to connect tools"</i>

CLUE Chatbot answers questions → GenAI app. **System decides, calls tools, and finishes the job** → agent. If a human has to do the last step, it's probably not agentic.

TRAP An agent isn't a different kind of model. It's an FM plus tools, memory, and a loop. Same brain, more hands.

BUSINESS USE Where agents earn their keep

Customer service

Look up the order, check policy, issue the refund

IT and operations

Triage a ticket, run diagnostics, open the change request

Research and analysis

Gather sources, compare, produce a briefing

FORMAT Expect a matching item here: concept on the left (memory, tools, orchestration, MCP), plain-English role on the right. Read the verbs. "Stores," "calls," "coordinates," "connects."

VISUAL Bedrock in one picture**What Bedrock gives you**

- **Model choice** through a single API
- **Knowledge Bases** for RAG on your own documents
- **Guardrails** to filter harmful or off-topic content
- **Prompt Management** to store and version prompts
- **Model evaluation** to compare models on your tasks
- **Customization**: fine-tuning, continued pre-training, distillation
- **Provisioned Throughput** for steady, high volume
- AWS states your inputs aren't used to train the base models

Bedrock vs. SageMaker AI

	Bedrock	SageMaker AI
Main job	Use existing FMs	Build, train, deploy your own ML
Infrastructure	Fully managed	You choose and manage more
Skill needed	Lower	ML engineering
Best for	Fast GenAI apps	Custom models, full control
Clue	"no ML infrastructure," "multiple FMs"	"build and train," "custom model"

AWS AI services you need to place

Service	What it is	Exam clue
Amazon Bedrock	Managed access to foundation models with RAG, guardrails, and customization	"build GenAI apps, no infrastructure"
Amazon SageMaker AI	Full platform to build, train, and deploy ML models	"custom ML model," "data scientists"
SageMaker JumpStart	A hub of pre-trained models and solution templates you can deploy quickly	"start from a pre-trained model"
Amazon Nova	Amazon's own family of foundation models, available through Bedrock	"Amazon's FMs"
Amazon Quick	Agentic AI workspace for business users. Bundles research, BI (Quick Sight, formerly QuickSight), and automation (Flows) over company data	"business users, no code"
Bedrock AgentCore	Platform to build, deploy, and operate agents securely at scale, with any framework or model	"run agents in production"
Strands Agents	Open-source SDK for building AI agents	"open-source agent framework"
Kiro	Agentic IDE and CLI built on spec-driven development: prompts become specs, then code	"spec-driven AI coding"
AWS Transform	Agentic AI service for modernizing and migrating legacy applications and infrastructure	"modernize .NET, mainframe"
Amazon Q	Added to scope in v1.1 and named in objective 1.3.4. AWS's generative AI assistant brand. The guide gives no depth, so recognize the name and role only.	name and role only

Why use AWS's managed GenAI services?

- Lower barrier to entry. No ML team required
- Faster to market
- Pay-as-you-go instead of buying hardware
- Security, compliance, and scaling come built in

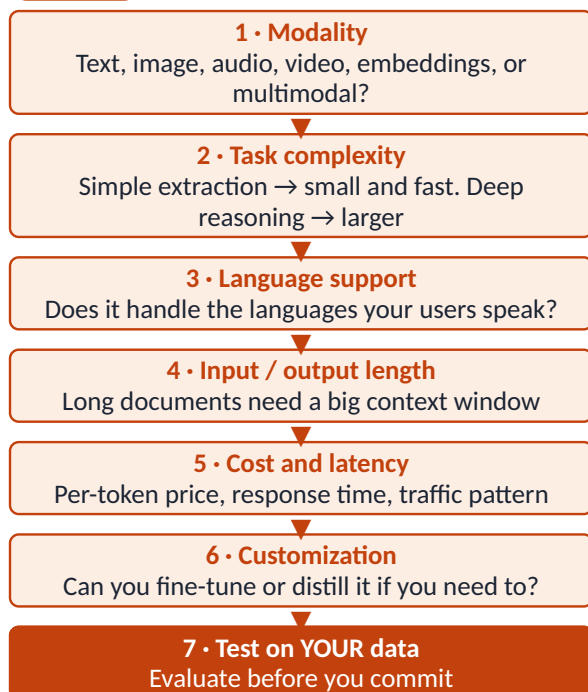
Tradeoffs to weigh

- **Cost**: on-demand tokens vs. Provisioned Throughput
- **Region availability**: not every model is in every Region
- **Customization**: custom models add cost and effort
- **Responsibility**: you still own your data, prompts, and access

CLUE "Build a GenAI app **without managing infrastructure**" or "**access multiple FMs through one service**" = Bedrock. "**Build and train a custom model**" = SageMaker AI.

★ **MUST MEMORIZE** Prompt first, then RAG, then fine-tune. Facts change → RAG. Behavior and style → fine-tuning.

VISUAL 8 The selection walk



What each criterion really means

Criterion	What to check	Exam clue
Cost	Price per input and output token. Smaller models cost less	" <i>minimize cost</i> "
Latency	Model size and answer length. Interactive apps need quick models	" <i>fastest response</i> "
Modality	The data types it can take in and put out	" <i>images and text</i> "
Language	Multilingual coverage and quality	" <i>users in many languages</i> "
Model size	Bigger = stronger reasoning, but slower and pricier	" <i>complex reasoning</i> "
Customization	Fine-tuning, continued pre-training, or distillation allowed	" <i>adapt to our domain</i> "
Input length	Context window must fit your prompt plus retrieved text	" <i>long contracts</i> "
Output length	Max tokens it can return. Long outputs add cost and delay	" <i>lengthy report</i> "
Performance	Benchmarks help, but your own test set decides	" <i>best quality for our task</i> "
Prompt caching	Reuses a repeated prompt prefix so it's cheaper and quicker	" <i>same long context every call</i> "

Smaller model vs. larger model

Pick smaller when	Pick larger when
Task is narrow: classify, extract, route	Task needs nuance and multi-step reasoning
Latency and cost matter most	Quality matters more than price
Traffic is huge	Volume is modest
You'll fine-tune or distill anyway	You want strong results out of the box

On-demand vs. Provisioned Throughput

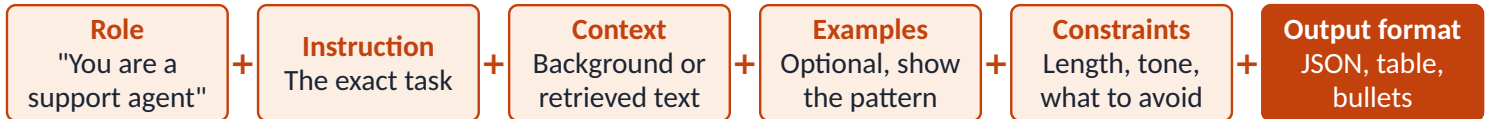
On-demand	Provisioned
Pay per token as you go	Reserve capacity for a set term
Spiky or unknown traffic	Steady, predictable, heavy traffic
Experiments and pilots	Production at scale, custom models

CLUE "Lowest latency and lowest cost for a simple task" → **smaller model**. "Strongest reasoning, cost is secondary" → **larger model**. "Compare candidate models fairly" → **run an evaluation on your own data**.

TRAP The "best" model on a public leaderboard isn't automatically the best fit. Context window, language, price, and latency can all knock it out.

Sanity check: three quick scenarios (made-up, not exam questions)

Situation	Lean toward	Why
High-volume support bot, short answers, users waiting	Smaller, fast model, low temperature	Latency and cost dominate
Summarizing very long contracts	Large context window	Input length is the constraint
Generating product images from text	Image-generation (diffusion) model	Output modality decides

VISUAL 9 Anatomy of a strong prompt**Zero-shot vs. one-shot vs. few-shot**

Style	Examples in prompt	What it looks like	Use when
Zero-shot	None	"Classify this review as positive or negative: <i>The battery died in a day.</i> "	Simple, common tasks
One-shot	One	Same request, plus one labeled sample review above it	A single example clears up the format
Few-shot	Several	Same request, plus three or more labeled samples	Custom formats, tone, or edge cases

CLUE "Provide examples **inside the prompt** without changing the model" = **in-context learning** (few-shot). The model's weights never change.

Techniques and what each one buys you

Technique	Effect
Role prompting	Sets voice and perspective
Instruction prompting	Clear, specific direction beats vague asks
Delimiters	Separate instructions from data ("" or XML-style tags)
Structured output	Ask for JSON or a table so software can use it
Chain-of-thought	"Reason step by step" improves multi-step answers
Negative prompting	Says what to avoid or leave out
Prompt templates	Reusable prompts with fill-in variables

Best-practice checklist

- Be specific and concise. Skip the fluff
- Say the audience, tone, and length
- Give examples when format matters
- Put long reference text in clearly marked blocks
- Define the output format explicitly
- Test, compare, tweak. Prompt optimization is iterative
- Add guardrails for anything user-facing

TRAP **Few-shot isn't fine-tuning.** Few-shot lives in the prompt and disappears after the call. Fine-tuning changes the model itself.

COACH TIP Prompting is always your **first** and **cheapest** move. If a question asks for the lowest-effort way to improve output, the answer is usually a better prompt.

CLUE "Model should **explain its reasoning** or work through a multi-step problem" → **chain-of-thought**. "Same prompt reused with different inputs" → **prompt template**. "Must return machine-readable results" → **structured output**.

Before and after: same task, better prompt

WEAK "Write about our return policy."

STRONG "You are a support agent for an online shoe store. Using the text inside <policy> tags, answer the customer's question in under 80 words, friendly tone. If the policy doesn't cover it, say so. Return JSON with keys answer and confidence."

What changed: a role, grounding context, a length limit, a fallback rule ("say so"), and a defined output format.

Temperature

LOW (near 0)

HIGH (near 1)

	Low	High
Output	Predictable, focused	Varied, creative
Repeatable?	Mostly	Rarely
Use for	Facts, extraction, code, support answers	Brainstorming, marketing copy, stories

CLUE "Make responses **more consistent / deterministic**" → **lower temperature**. "More creative or diverse responses" → **raise temperature**.

Input and output length

- **Input length** = how much you send. Must fit the context window
- **Output length** (max tokens) = the ceiling on the response
- Cap it too low and answers get **cut off mid-sentence**
- Longer in or out = **higher cost** and **slower replies**

Other sampling dials like top-p and top-k work the same way: tighter means safer and more predictable.

TRAP Temperature ≠ context window. Temperature controls *randomness*. The context window controls *how much text fits*. Different knobs, different problems.

Prompt risks: attacks on your GenAI app

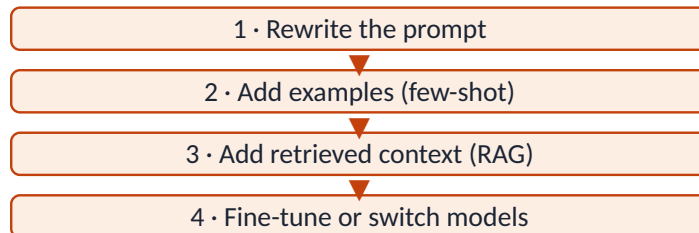
Risk	What's happening	Main defense
Prompt injection	An attacker hides instructions in the input ("ignore your rules and...")	Guardrails, input validation, separate instructions from data
Hijacking	The model is steered into doing something the attacker wants instead of the intended task	Least-privilege tools, human approval for risky actions
Jailbreaking	Tricking the model into bypassing its safety rules	Guardrails and content filters, red-team testing
Poisoning	Bad data planted in training or retrieval sources to skew results	Vet data sources, control who can write to them
Exposure	Sensitive data or hidden instructions leak through prompts or outputs	Don't put secrets in prompts; mask PII; filter outputs

COACH TIP Treat retrieved documents and user text as **untrusted input**. If an agent can take actions, give it the minimum permissions it needs and log everything it does.

Amazon Bedrock Prompt Management

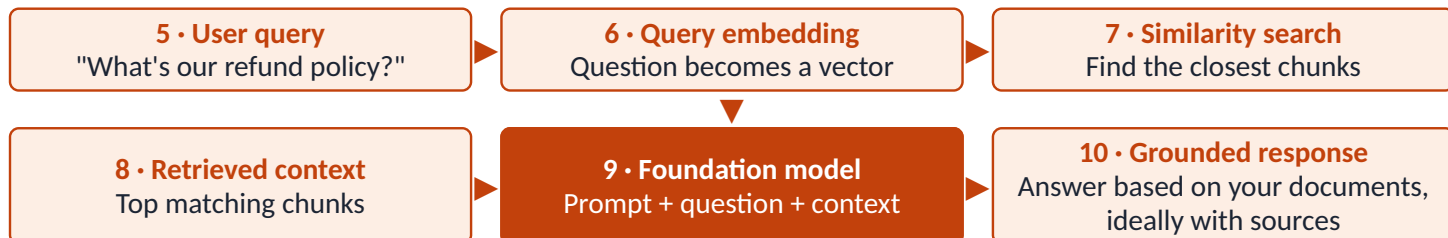
- Create, save, and **version** prompts as managed assets
- Use **variables** so one prompt serves many inputs
- Test prompts against different models
- Keeps teams consistent instead of copy-pasting prompts around

When the model's answer is off, climb this ladder



Settings in practice

Task	Temperature	Output length
Extract fields from an invoice	Low	Short
Answer from a policy document	Low	Short to medium
Generate or fix code	Low	As needed, cap it
Summarize a report	Low to medium	Medium
Brainstorm product names or taglines	High	Short lists

VISUAL 5 RAG: two phases, ten steps**PHASE 1 · Prepare your knowledge (ongoing, before anyone asks)****PHASE 2 · Answer a question (every request)****What RAG solves**

- **Stale knowledge.** The model's training has a cutoff
- **Missing private data.** It never saw your documents
- **Hallucinations.** Answers are anchored to retrieved text
- **Trust.** You can show the source of each answer

Why it reduces hallucination risk

Instead of asking the model to recall facts from memory, you hand it the relevant passages and tell it to answer from those. It's the difference between a closed-book and an open-book test. It lowers the risk. It doesn't erase it, so you still check outputs.

RAG or fine-tuning? Use RAG when...

Use RAG	Reach for fine-tuning instead
Facts change often (policies, prices, product docs)	Knowledge is stable, behavior is what you want to change
You need answers tied to specific documents	You need a particular tone, format, or specialized style baked in
You want fast, cheaper updates: re-index, no retraining	You have a quality labeled dataset and budget to train
You need source citations or audit trails	The task is the same narrow pattern repeated at volume

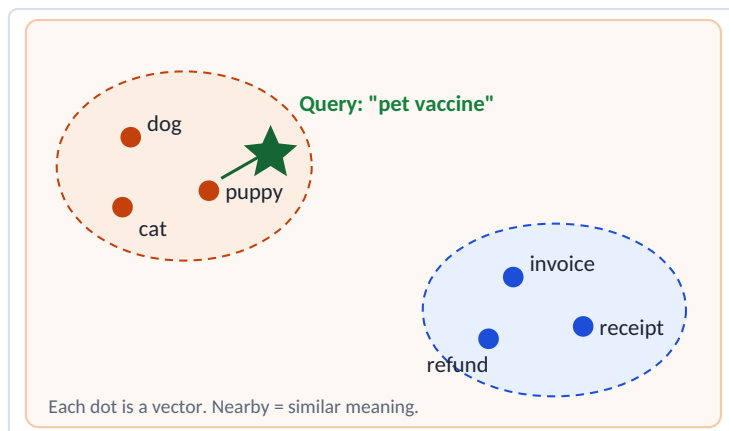
CLUE "Ground responses in **enterprise documents**" or "answers must reflect **current company data**" → **RAG**. On AWS, the managed path is **Knowledge Bases for Amazon Bedrock**.

TRAP **RAG doesn't change the model.** Nothing is retrained. The model just gets better material in the prompt. And it only works as well as the retrieval: bad chunks in, weak answer out.

FORMAT An **ordering** item may ask you to sequence these steps. Anchor on this: **chunk → embed → store** happens first; **query → embed → search → augment → generate** happens at question time.

Where RAG goes wrong, and how to fix it

Symptom	Likely cause	Fix
Ignores your documents	Wrong chunks retrieved	Better chunking, better embedding model, metadata filters
Half-right answers	Chunks too small, context lost	Adjust chunk size and overlap
Still makes things up	Prompt doesn't force grounding	"Answer only from the context," citations, grounding checks
Outdated answers	Index is stale	Re-sync the data source
Shows restricted content	Retrieval ignores permissions	Filter results by user access

VISUAL 6 Embedding → vector → similarity search1 • **Embedding model** turns text/images into vectors2 • **Vector database** stores vectors + metadata3 • **Query** is embedded the same way4 • **Similarity search** returns the nearest vectors

5 • Matching content goes to the FM as context

TRAP The **embedding model** and the **text-generation model** are two different models doing two different jobs.

AWS vector store options the guide names

Option	What it is	Pick it when
Amazon OpenSearch Service	Search and analytics engine with vector search built in	You want search features, scale, or keyword + semantic together
Amazon Aurora	Managed relational database (PostgreSQL-compatible) with vector support	Your data already lives in Aurora and you want vectors beside it
Amazon RDS for PostgreSQL	Managed PostgreSQL with a vector extension	You're already on RDS PostgreSQL and need a simple setup
Amazon Neptune	Graph database that can also do vector search	Relationships between things matter (knowledge graphs)

Vector database vs. traditional database

	Vector	Traditional
Finds by	Meaning (closeness)	Exact values, filters
Stores	Embeddings + metadata	Rows, documents
Question it answers	"What's similar?"	"What matches exactly?"

CLUE "Find content by **meaning**, not keywords" or "semantic search" → **embeddings + a vector database**.

Knowledge Bases for Amazon Bedrock

The managed shortcut to RAG. Point it at your data (for example, documents in S3). It chunks, embeds, and stores them in the vector store you choose, then retrieves relevant text at question time and passes it to the model. Responses can include **source citations**.

TRAP MemoryDB was dropped from the current in-scope list. Stick to the four above for vector-store questions.

Embedding gotchas

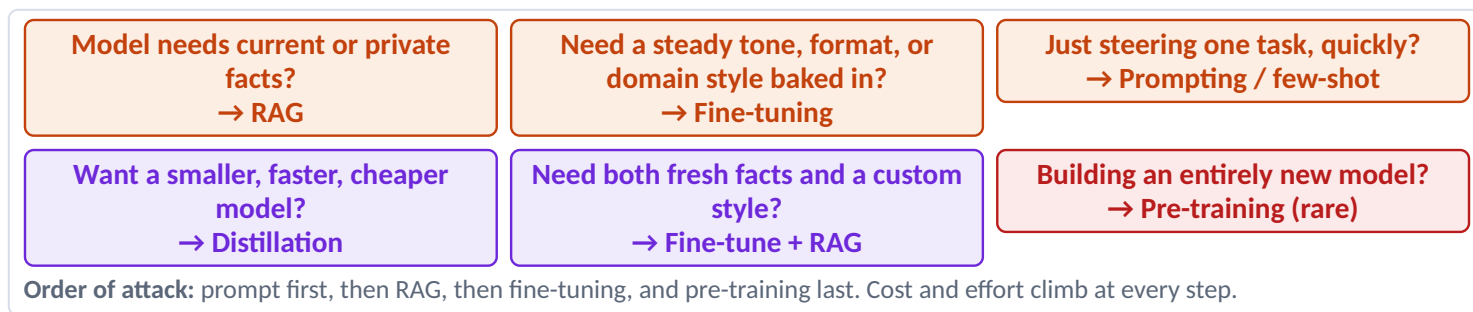
- Use the **same embedding model** for documents and queries
- Switch embedding models and you must **re-embed everything**
- **Small chunks** are precise but lose context. **Large chunks** keep context but add noise
- **Metadata** (source, date, owner) powers filters and citations
- Keep the vector store **in sync** with the source documents

Embeddings beyond RAG

- Semantic search
- Recommendations ("more like this")
- Clustering similar items
- Classifying text by meaning
- Spotting duplicates and near-duplicates

MASTER TABLE In-context learning vs. RAG vs. fine-tuning vs. distillation vs. pre-training

Approach	What changes	Data	Cost / effort	Use when	Exam clue
In-context learning (few-shot)	Only the prompt. Model untouched	A few examples	Lowest	Quick steering for one task	<i>"examples in the prompt"</i>
RAG	What the model sees at answer time. Model untouched	Your documents, indexed	Low to moderate	Current, private, or citable knowledge	<i>"ground in company documents"</i>
Fine-tuning	The model's weights, for a specific task or style	Labeled examples (prompt + ideal response)	Moderate to high	Consistent tone, format, or domain behavior	<i>"organization-specific behavior"</i>
Distillation	A small model is trained to imitate a big one	Outputs from the large model	Moderate	Need big-model quality at small-model cost and speed	<i>"smaller, cheaper, similar quality"</i>
Pre-training	Everything. Model is built from scratch (or continued on new unlabeled data)	Massive unlabeled data	Highest	Almost never. Only to build a new FM or deeply absorb a domain's language	<i>"from scratch," "huge corpus"</i>

VISUAL 7 Decision tree: which one do I need?**Ways to fine-tune (3.3)**

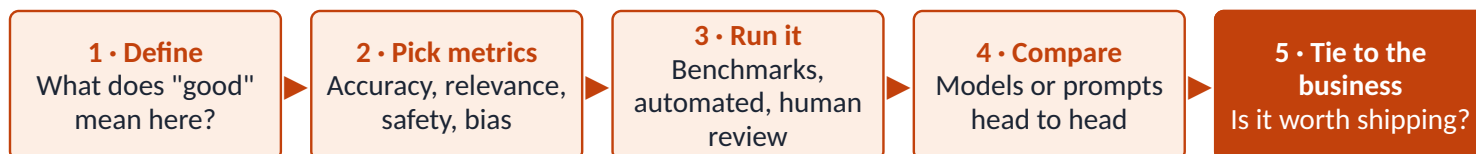
Method	What it does
Instruction tuning	Teaches the model to follow your style of instructions
Domain adaptation	Tunes it to a field like legal or medical
Transfer learning	Starts from a trained model and adapts it to a new task
Continued pre-training	Keeps training on unlabeled domain data
RLHF	Humans rank outputs to steer the model toward preferred responses

Preparing fine-tuning data

- **Curate** it: accurate, relevant, representative
- **Label** it: prompt and ideal-response pairs
- **Size** it: enough examples to teach the pattern
- **Govern** it: check privacy, licensing, and bias
- Bad data in, bad model out. Fine-tuning amplifies flaws

TRAP Fine-tuning isn't the fix for stale facts. It changes behavior, not a live knowledge feed. Facts change? **RAG**.

CLUE "Lowest cost and effort" → **prompting**. "Without retraining" → **RAG** or **in-context learning**. "Retrain on labeled data" → **fine-tuning**.

VISUAL 14 The evaluation framework

Human vs. automated evaluation			Named metrics and benchmarks	
	Human	Automated	Metric	Measures
Strength	Judges tone, nuance, helpfulness	Fast, cheap, repeatable	ROUGE	Overlap with a reference. Summarization
Weakness	Slow, costly, subjective	Can miss subtle quality	BLEU	Similarity to reference translations
Use for	Final quality calls, subjective traits	Large-scale comparisons, regression checks	BERTScore	Semantic similarity using embeddings
LLM-as-a-judge uses a second model to score outputs. It scales like automation but still needs spot checks from people.			Benchmark datasets	Standard test sets for comparing models

CLUE CHART Which problem is being described?

The question says...	Think	Typical fix
Answers contain claims that aren't in the source	Groundedness / accuracy	RAG, citations
Answers wander off the question asked	Relevance	Better prompt, better retrieval
Output includes offensive or harmful language	Toxicity	Guardrails
Results are worse for certain groups	Bias / fairness	Representative data, subgroup tests
A typo or odd phrasing breaks the answer	Robustness	Test with messy inputs
Compare a summary against a human-written reference	ROUGE	Automated scoring
Judge tone, brand voice, or "does this feel right?"	Human evaluation	Reviewer panel

Business alignment metrics (new in v1.1)

- **Task completion rate**: did it finish the job?
- **User satisfaction** and feedback scores
- **Cost per interaction** and overall ROI
- **Productivity** gains and conversion impact

Evaluating RAG and agents

- **RAG**: judge the *retrieval* (right chunks?) and the *answer* (faithful to them?)
- **Agents**: right tool, right steps, task actually completed, safe actions
- Amazon Bedrock offers **model evaluation** for comparing models

TRAP A strong benchmark score doesn't prove business value. If the exam asks whether the solution *met business goals*, look for completion rate, satisfaction, or cost metrics, not just accuracy.

Which evaluation approach for which situation?

Situation	Best fit
Quickly compare three models on summarization	Automated metrics (ROUGE, BERTScore)
Judge brand voice and tone	Human evaluation
Score thousands of chatbot answers every night	LLM-as-a-judge plus human spot checks
Safety sign-off before launch	Toxicity and bias evaluations plus human review

★ **MUST MEMORIZE** Guardrails shape what the model says. IAM controls who can call it. Bias is not hallucination.

VISUAL 10 The Responsible AI framework

FAIRNESS Similar outcomes for similar people, across groups

INCLUSIVITY Works for diverse users and needs

ROBUSTNESS Holds up under odd, noisy, or hostile input

SAFETY Doesn't cause harm to people or society

VERACITY Outputs are truthful and accurate

TRANSPARENCY Open about what it is, how it was built, and its limits

EXPLAINABILITY You can say why it produced an output

PRIVACY Protects personal and sensitive data

HUMAN OVERSIGHT People review and can override high-stakes decisions

SUSTAINABILITY Mind the energy and resources a model burns

GOVERNANCE Clear policies, owners, and review for how AI gets built and used

CHECKLIST Before you ship: ask these eight questions

Question	What a "yes" looks like
Fair?	Tested across demographic groups. No group is consistently worse off
Safe?	Harmful output blocked. Guardrails on. Humans review high-risk calls
Accurate?	Grounded in trusted sources. Hallucinations measured and reduced
Transparent?	Users know it's AI. Data sources and limits are documented
Explainable?	You can justify decisions to a customer, auditor, or regulator
Private?	Sensitive data masked, access-controlled, and encrypted
Robust?	Stays reliable on messy inputs and attempted manipulation
Inclusive?	Training data and testing reflect the people who'll use it

Responsible AI risks: meaning and mitigation

Risk	What it means	Typical mitigation
Bias	Unfair outcomes for certain groups	Representative data, subgroup evaluation
Hallucination	Confident, false output	Grounding with RAG, human review
Toxicity	Harmful or offensive output	Guardrails and content filtering
Privacy exposure	Sensitive data leaks in or out	Access controls, masking, data protection
IP infringement	Output echoes copyrighted or licensed work	Data and model governance, license checks
Poor explainability	Nobody can say why it decided that	Explainability tools, simpler models, documentation

What good datasets look like (4.1.3)

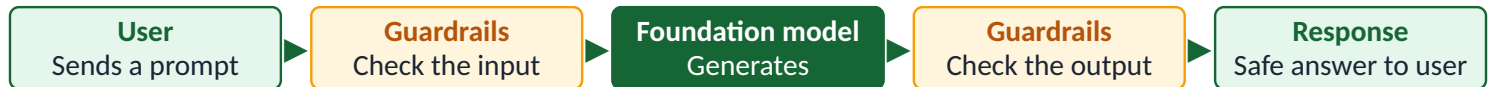
- **Diverse** and **inclusive**
- **Balanced** across groups and classes
- **Curated** from trusted, documented sources
- **Representative** of real users

Biased data in, biased model out. This is the root cause the exam keeps circling back to.

Legal risks of GenAI (4.1.4)

- IP infringement claims
- Biased outputs that create liability
- Loss of customer trust
- Harm to end users
- Hallucinations that mislead people

CLUE Wording like "fair across demographic groups" → fairness/bias. "Works for all users" → inclusivity. "Reliable under unexpected input" → robustness. "Correct and truthful" → veracity.

VISUAL 11 Amazon Bedrock Guardrails: a checkpoint on both sides**What Guardrails can do**

Control	Effect
Content filters	Block harmful content such as hate, insults, violence, and prompt attacks
Denied topics	Keep the model off subjects you define
Word filters	Block specific words or phrases
Sensitive information filters	Detect and mask or block PII
Grounding checks	Flag answers not supported by the source

Guardrails vs. IAM

	Guardrails	IAM
Controls	What the model may say	Who may <i>call</i> AWS resources
Layer	Responsible AI	Security
Example	Block medical advice	Only this role can invoke the model

TRAP: RESPONSIBLE AI VS. SECURITY Guardrails shape **behavior**. IAM, KMS, and PrivateLink protect **access and data**. A question about harmful output is not an IAM question.

Transparency vs. explainability

	Transparency	Explainability
Question	"How was this built, with what data, and what are its limits?"	"Why did it give this answer?"
Delivered by	Documentation, Model Cards, disclosures	Feature importance, explanation tools
Audience	Users, auditors, regulators	Anyone reviewing a decision

TRAP These two get swapped constantly. Documentation about the model = transparency. Reasons for one output = explainability.

AWS tools to know here

Tool	Use it to...
SageMaker Clarify	Detect bias and explain model predictions
SageMaker Model Cards	Document a model's purpose, data, and limits
Bedrock Guardrails	Filter harmful or unwanted content
Bedrock model evaluation	Check quality, bias, and safety across models
Human review	Add oversight where stakes are high

Interpretable models vs. black boxes

Simple models (linear, decision tree) are easy to explain but may be less accurate. Complex deep-learning models perform better but are harder to explain. When regulators or customers demand reasons, **that tradeoff** is the answer.

Open-source vs. proprietary models

Open models offer more visibility into the architecture. Proprietary models are often stronger out of the box but less transparent. Check licensing either way. It's a governance question as much as a technical one.

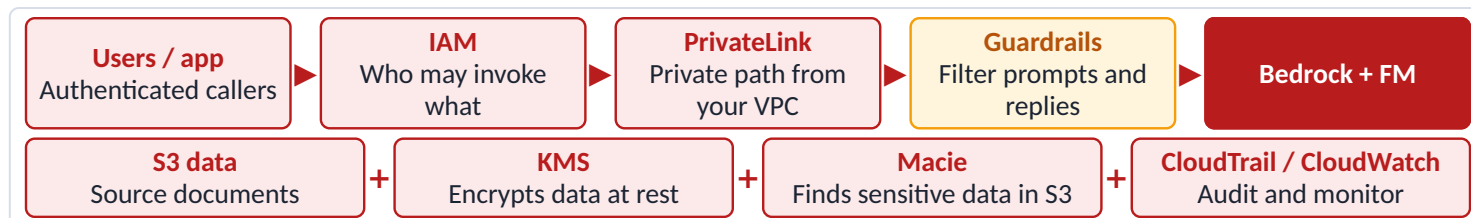
CLUE "Reduce harmful or unwanted responses" → Guardrails. "Document the model and its training data" → Model Cards. "Detect bias / explain predictions" → SageMaker Clarify.

Staying responsible after launch

Monitor Watch outputs, quality, and bias over time	Collect feedback Let users flag bad answers; route high-risk ones to humans	Re-evaluate Retest on fresh data, update guardrails, keep audit logs
--	---	--

★ **MUST MEMORIZE** AWS secures the cloud. You secure what you put in it. Least privilege, always.

VISUAL 13 An AI security architecture, layer by layer



SHARED RESPONSIBILITY Who secures what

AWS: SECURITY OF THE CLOUD Data centers, hardware, network, the managed-service infrastructure, and hosting the models

YOU: SECURITY IN THE CLOUD Your data, prompts, IAM permissions, encryption choices, guardrail settings, app code, and how you use the output

Managed service, same rule. Bedrock removes servers from your plate, not accountability for your data and access.

IAM in five lines

- **Users:** individual identities
- **Roles:** temporary identities for people, apps, or services
- **Policies:** documents that allow or deny actions
- **Permissions:** what a policy grants
- **Least privilege:** only what the job needs, nothing extra

KMS vs. IAM

KMS	IAM
Creates and manages encryption keys	Controls who can do what
"encrypt data at rest"	"restrict access"

Macie vs. IAM

Macie	IAM
Discovers sensitive data in S3	Grants or denies access
"find PII"	"who can open it"

Guardrails vs. IAM

Guardrails	IAM
Filters model content	Gates API access
"block toxic output"	"who can invoke"

Encryption and data protection

Idea	What it means
At rest	Data is encrypted where it's stored (S3, databases, vector stores)
In transit	Data is encrypted as it moves (TLS)
KMS	The service that creates and controls the keys
Access control	Only approved identities touch the data
Data integrity	Data stays accurate and unaltered

Amazon Macie and AWS PrivateLink

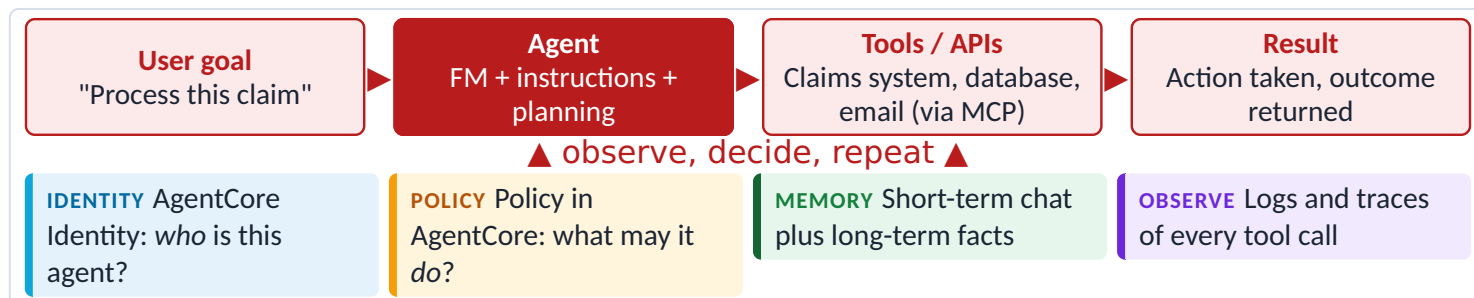
Macie uses machine learning and pattern matching to find sensitive data, like PII, in **Amazon S3**. When a question says "identify and protect sensitive data before it feeds an AI workload," think Macie.

PrivateLink keeps traffic between your VPC and an AWS service on the AWS network instead of the public internet. For AI, that means calling Bedrock without sending prompts over the open internet.

CLUE "Control who can access the model or data" → **IAM**. "Encrypt" → **KMS**. "Find sensitive data in S3" → **Macie**. "Private connection, no public internet" → **PrivateLink**.

Data protection checklist for AI workloads

1. Classify your data first
2. Discover sensitive data (Macie)
3. Send the model only what it needs
4. Encrypt at rest and in transit
5. Restrict access with least privilege
6. Log activity (CloudTrail)
7. Keep data in approved Regions
8. Delete what you no longer need

VISUAL 15 An agentic AI architecture, with the controls marked**Agent security: which control answers which question?**

The worry	Control	Why
Which workload or agent is making this request?	AgentCore Identity	Gives agents their own verifiable identity and credentials
What may a permitted agent do with its tools?	Policy in AgentCore	Fine-grained rules that limit actions, enforced on tool calls
Who may call an AWS API at all?	IAM	Account-level access control
What may the model say?	Bedrock Guardrails	Content and topic controls
What did the agent actually do?	Logging and audit trails	Accountability after the fact

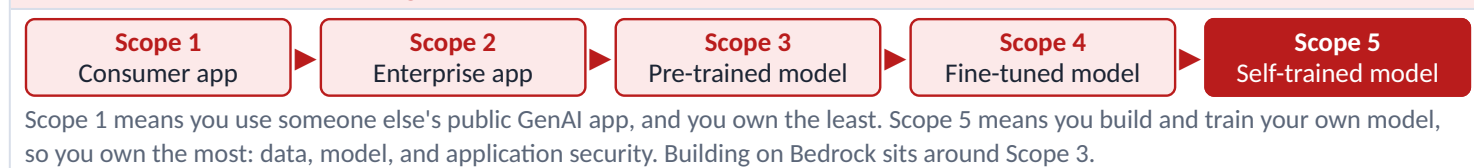
TRAP Identity answers "who." Policy answers "what's allowed." An authenticated agent with no policy can still do everything its credentials permit, which is usually far more than the task needs. Give agents **least privilege**.

Threats unique to GenAI

- **Prompt injection** and jailbreaks
- **Data leakage** through prompts or outputs
- **Poisoned data** in training or retrieval sources
- **Over-permissioned agents** taking risky actions
- **Hallucinated** content treated as fact

Hallucination detection and grounding (5.1.5)

- **Ground** answers in trusted sources (RAG)
- **Cite** the source for each claim
- Run **grounding checks** (Guardrails)
- **Evaluate** faithfulness before launch
- Keep a **human in the loop** for high stakes

Generative AI Security Scoping Matrix: the more you build, the more you own

CLUE "Which controls **an agent's access** to tools and what it may do" → **agent identity and policy**. "How much **security responsibility** do we carry depending on how we use GenAI?" → **Security Scoping Matrix**.

Defense in depth for a GenAI app

AWS services that support compliance and governance

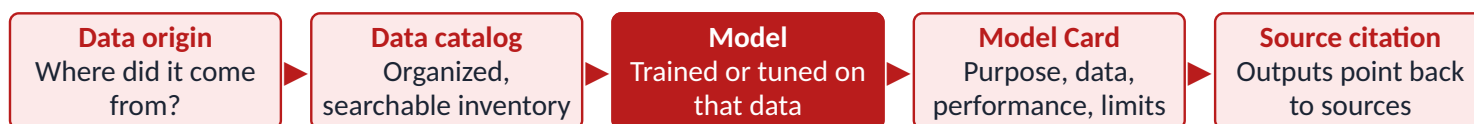
Service	What it does	Exam clue
AWS Artifact	On-demand access to AWS compliance reports and agreements	"download AWS audit reports"
AWS Config	Tracks resource configurations and checks them against rules	"is it configured to policy?"
Amazon Inspector	Scans workloads for software vulnerabilities and unintended exposure	"find vulnerabilities"
AWS CloudTrail	Records API calls: who did what, when	"audit trail"
AWS Trusted Advisor	Best-practice checks across cost, performance, security, and limits	"recommendations to improve the account"
Amazon CloudWatch	Metrics, logs, and alarms for monitoring	"monitor and alert"

Governance in practice (5.2.2)

- **Policies** that define acceptable AI use
- **Review cadence** for models and data
- **Strategies** for risk and change management
- **Transparency standards** and documentation
- **Team training** on responsible AI

Data governance strategies

- **Data lifecycle**: collect, use, retain, delete
- **Logging** and **monitoring** of access
- **Residency**: data stays in required Regions
- **Retention** rules that match regulations
- **Quality and integrity** checks

VISUAL Data lineage and documentation: the paper trail

Why documenting sources matters

- **Traceability**: you can follow an output back to its data
- **Audits**: prove what the model was built from
- **Bias investigations**: find where skew entered
- **Licensing and copyright**: confirm you're allowed to use it
- **Trust**: users can verify claims

The terms, side by side

Term	Meaning
Data lineage	The full path data travels, from source through every transformation
Data origin	Where the data first came from
Data cataloging	Inventorying data so it can be found and governed
Source citation	Naming the source behind an answer
Model Card	A standard document describing a model (SageMaker Model Cards)

Compliance and regulation: what the exam wants you to recognize

Regulated industries (finance, healthcare, government) face rules on data privacy, explainability, and audits. AI-specific regulations exist and are growing, and algorithm accountability is part of that conversation. You won't be asked about specific laws. You *will* be asked which practices help you meet them: documentation, audit logs, access control, data governance, and human oversight.

CLUE "Need **compliance reports** from AWS" → **Artifact**. "Track **configuration changes**" → **Config**. "Record **who called the API**" → **CloudTrail**. "Document **model and data details**" → **Model Cards**.

TRAP **CloudTrail** ≠ **CloudWatch**. CloudTrail records *who did what* (API activity). CloudWatch watches *how things are running* (metrics, logs, alarms).

DOMAIN 1 • AI & ML fundamentals

- 1 AI $\supset$ ML $\supset$ deep learning $\supset$ GenAI. Agents act with tools.
- 2 Training builds the model. Inference uses it.
- 3 Supervised = labeled. Unsupervised = not. RL = rewards.
- 4 Classification = category. Regression = number.
- 5 Clustering = grouping unlabeled data.
- 6 Real-time = someone's waiting. Batch = offline bulk.
- 7 Async = big or slow jobs. Serverless = spiky traffic.
- 8 Structured = rows and columns. Unstructured = text, image, audio.
- 9 Recommendations $\rightarrow$ Personalize. Scanned forms $\rightarrow$ Textract.
- 10 Images and video $\rightarrow$ Rekognition. Sentiment $\rightarrow$ Comprehend.
- 11 Overfit = great on training, poor on new (high variance).
- 12 Underfit = poor on both (high bias).
- 13 Judge a model on the test set, never training data.
- 14 Costly misses $\rightarrow$ recall. Costly false alarms $\rightarrow$ precision.
- 15 Accuracy misleads on lopsided data. F1 balances P and R.
- 16 Order: collect, EDA, prep, features, train, tune, test, deploy, monitor.
- 17 MLOps = repeatable, monitored, retrainable ML.
- 18 Explainability or tight limits $\rightarrow$ traditional ML.
- 19 Exact, rule-based answer needed? Skip ML.
- 20 ROI and cost per user decide if a model was worth it.

DOMAIN 2 • Generative AI

- 21 FM = broad pre-training, many tasks. LLM = language.
- 22 Tokens = text chunks. They drive cost and limits.
- 23 Context window = max tokens per request, in plus out.
- 24 Embeddings = meaning as vectors. Near = similar.
- 25 Transformers use attention to weigh word relevance.
- 26 Diffusion models create images from noise.
- 27 Multimodal = text, image, audio, video together.
- 28 LLMs predict the next token. They don't look facts up.
- 29 Hallucination = confident, false. Fix: RAG, citations, review.
- 30 Want consistent output? Lower the temperature.
- 31 Toxic output $\rightarrow$ Guardrails.
- 32 Bias = skewed data. Hallucination = ungrounded guess.
- 33 You pay for input and output tokens.
- 34 Cut cost: shorter prompts, smaller model, caching.
- 35 Provisioned Throughput = reserved capacity, steady load.
- 36 Context engineering = choosing what the model sees.
- 37 Agent = FM + instructions + tools + memory + loop.
- 38 MCP = open standard connecting agents to tools.
- 39 Multi-agent = specialists, often with a supervisor.
- 40 Bedrock = many FMs, no ML infrastructure.
- 41 SageMaker AI = build, train, deploy custom ML.
- 42 JumpStart = pre-trained models and templates.
- 43 AgentCore = run agents securely at scale.
- 44 Strands = agent SDK. Kiro = spec-driven agentic IDE. Transform = modernization.

DOMAIN 3 • Foundation model applications

- 45 Pick a model: modality, task, language, context, cost, latency.
- 46 Smaller = cheaper, faster. Larger = stronger reasoning.
- 47 Prompt caching cheapens repeated prompt prefixes.
- 48 Test candidate models on your own data.
- 49 Zero-shot = 0 examples. One-shot = 1. Few-shot = several.
- 50 Few-shot is in-context learning. Model unchanged.

DOMAIN 3 • Foundation model applications (cont.)

- 51 Chain-of-thought = step-by-step reasoning.
- 52 Templates and delimiters keep prompts clean and reusable.
- 53 Software consuming the output? Ask for JSON.
- 54 Higher temperature = varied. Lower = predictable.
- 55 Temperature = randomness. Context window = capacity.
- 56 Output caps affect cost, speed, truncation.
- 57 Prompt risks: injection, jailbreak, hijack, poison, exposure.
- 58 Prompt Management stores and versions prompts.
- 59 RAG: chunk, embed, store; query, embed, search, generate.
- 60 RAG grounds answers in your docs. No retraining.
- 61 Facts change $\rightarrow$ RAG. Style and behavior $\rightarrow$ fine-tuning.
- 62 Knowledge Bases for Bedrock = managed RAG.
- 63 Semantic search $\rightarrow$ embeddings + vector database.
- 64 Vector stores: OpenSearch, Aurora, RDS PostgreSQL, Neptune.
- 65 Neptune fits when relationships matter.
- 66 Fine-tuning changes weights with labeled examples.
- 67 Distillation: a small model mimics a big one.
- 68 Pre-training from scratch: rare and costliest.
- 69 Escalate: prompt, RAG, fine-tune, pre-train.
- 70 RLHF aligns models using human feedback.
- 71 ROUGE = summaries. BLEU = translation. BERTScore = meaning.
- 72 Humans judge nuance. Automation scales. LLM-judge needs spot checks.

DOMAIN 4 • Responsible AI

- 73 Pillars: fairness, inclusivity, robustness, safety, veracity, privacy.
- 74 Biased output usually starts with unrepresentative data.
- 75 Good data: diverse, balanced, curated, representative.
- 76 Guardrails: harmful content, denied topics, PII, ungrounded answers.
- 77 Guardrails shape behavior. IAM controls access.
- 78 Transparency = documentation. Explainability = why this output.
- 79 Model Cards document purpose, data, and limits.
- 80 SageMaker Clarify = detect bias, explain predictions.
- 81 More complex models explain less.
- 82 High-stakes decisions need human oversight.
- 83 Legal risks: IP, bias, lost trust, user harm, hallucination.
- 84 Robustness = survives noisy or hostile input.
- 85 Veracity = truthful. Sustainability = resource use.
- 86 Responsible AI spans the whole lifecycle.

DOMAIN 5 • Security & governance

- 87 AWS secures the cloud. You secure what you put in it.
- 88 IAM = who can do what. Least privilege.
- 89 Roles give temporary credentials. Prefer them to keys.
- 90 KMS = encryption keys. Encrypt at rest and in transit.
- 91 Macie finds sensitive data like PII in S3.
- 92 PrivateLink keeps traffic off the public internet.
- 93 CloudTrail = who did what. CloudWatch = how it's running.
- 94 Artifact = reports. Config = rules. Inspector = vulnerabilities.
- 95 Trusted Advisor = best-practice checks.
- 96 AgentCore Identity = who. Policy = what's allowed.
- 97 Treat retrieved text and user input as untrusted.
- 98 Scoping Matrix: Scope 1 consumer to 5 self-trained. Build more, own more.
- 99 Lineage, origin, catalog, citations = auditable AI.
- 100 Governance: policies, review, lifecycle, residency, retention.

VISUAL 12 AWS AI service selection map

GenAI & agents		Pre-built AI (no ML skills)	
Amazon Bedrock	Use FMs through one API. Includes Knowledge Bases (RAG), Guardrails, Prompt Management	Rekognition	Images and video
Amazon Nova	Amazon's own FMs, on Bedrock	Textract	Text and fields from documents
Bedrock AgentCore	Build, deploy, run agents securely at scale	Comprehend	Sentiment, entities, topics
Strands Agents	Open-source agent SDK	Transcribe	Speech to text
Amazon Quick	Agentic workspace: research, BI (Quick Sight), automation	Polly	Text to speech
Kiro	Spec-driven agentic IDE and CLI	Translate	Language translation
AWS Transform	Agentic app and infrastructure modernization	Lex	Voice and text chatbots
		Personalize	Recommendations
Build your own ML		Recognition only	
SageMaker AI	Build, train, deploy custom models	Amazon Q	AWS's GenAI assistant brand. In v1.1 scope; recognition only
SageMaker JumpStart	Pre-trained models and templates	Kendra	Enterprise search use case
SageMaker Clarify	Bias detection and explainability	Fraud Detector	Fraud detection use case
SageMaker Model Cards	Model documentation	Forecast	Time-series forecasting use case
Vector stores	OpenSearch, Aurora, RDS PostgreSQL, Neptune	Out of scope	GuardDuty, WAF, Shield, Cognito, Route 53: skip

GUIDE Prompting → RAG → fine-tuning: how far do I need to go?**CLUES** Question clue → answer (GenAI and evaluation)

The question says...	Think	The question says...	Think
Make output more consistent	Lower temperature	Confident but wrong answer	Hallucination
More creative, varied output	Higher temperature	Same prompt, different answers	Non-determinism
Examples in the prompt, model unchanged	Few-shot / in-context	Worse results for one group	Bias
Show step-by-step reasoning	Chain-of-thought	Offensive or harmful output	Toxicity / Guardrails
Reusable prompt with fill-in variables	Prompt template	Score a summary against a reference	ROUGE
Machine-readable output	Structured output (JSON)	Score a translation	BLEU
Ground answers in enterprise documents	RAG	Judge tone and subjective quality	Human evaluation
Fresh facts without retraining	RAG	Score thousands of outputs	Automated / LLM-judge cheaply
Learn our tone or domain behavior	Fine-tuning	Which model is best for us?	Evaluate on your data
Smaller, cheaper, similar quality	Distillation	Steady, heavy traffic	Provisioned Throughput
Build a brand-new FM	Pre-training	Same long prefix on every call	Prompt caching
Find content by meaning	Embeddings + vector DB	Standard way for agents to reach tools	MCP
Pieces of text counted for billing	Tokens	Did it meet business goals?	Completion, satisfaction, cost
Maximum text per request	Context window	Missing a real case is costly	Recall

CLUES Question clue → answer (security, governance, responsible AI)

The question says...	Think
Control who can access AWS resources	IAM
Create and manage encryption keys	KMS
Find sensitive data in S3	Macie
Private path, no public internet	PrivateLink
Reduce harmful or unwanted responses	Bedrock Guardrails
Mask PII in prompts and replies	Guardrails filters
Document a model's purpose and data	Model Cards
Detect bias, explain predictions	SageMaker Clarify
Follow data from source to model	Data lineage
Name the source behind an answer	Source citation
Who made this API call?	CloudTrail
Is this resource configured to policy?	AWS Config

The question says...	Think
Download AWS compliance reports	AWS Artifact
Scan workloads for vulnerabilities	Amazon Inspector
Best-practice recommendations	Trusted Advisor
Which agent is calling?	AgentCore Identity
What may that agent do?	Policy in AgentCore
Give only the permissions needed	Least privilege
Who secures what, AWS or you?	Shared responsibility
How much security do we own?	Scoping Matrix
People review high-stakes calls	Human oversight
Must justify decisions to regulators	Explainability
Documentation of how it was built	Transparency
Stays reliable on hostile input	Robustness

⚠ Don't confuse these: 36 traps (★ marks the 30 highest-yield)

- ★ **RAG vs. fine-tuning.** RAG hands the model fresh facts at question time. Fine-tuning changes its behavior. Stale facts? RAG.
- ★ **Bedrock vs. SageMaker AI.** Bedrock uses ready FMs, no infrastructure. SageMaker AI builds and trains your own.
- ★ **Training vs. inference.** Building the model is training. Using it is inference, even if the question never says so.
- ★ **Embeddings vs. tokens.** Tokens are counted. Embeddings are compared.
- ★ **Temperature vs. context window.** Randomness versus capacity. Different problems.
- ★ **IAM vs. Guardrails.** IAM gates who calls. Guardrails shape what the model says.
- ★ **Bias vs. hallucination.** Unfair is bias. Made-up is hallucination.
- ★ **Explainability vs. transparency.** Why this output, versus how it was built and documented.
- ★ **Classification vs. regression.** Category versus number. Yes/no is classification.
- ★ **Few-shot vs. fine-tuning.** Few-shot lives in the prompt. Fine-tuning changes the weights.
- ★ **Precision vs. recall.** False alarms hurt: precision. Missed cases hurt: recall.
- ★ **Overfit vs. underfit.** Overfit memorized training data. Underfit is too simple.
- ★ **Accuracy on lopsided data.** 99% accurate can still be useless. Check precision, recall, F1.
- ★ **KMS vs. IAM.** KMS manages keys. IAM manages permissions.
- ★ **Macie vs. IAM.** Macie finds sensitive data. IAM grants access.
- ★ **CloudTrail vs. CloudWatch.** Who did it, versus how it's running.
- ★ **AgentCore Identity vs. Policy.** Who the agent is, versus what it may do.
- ★ **Agent vs. chatbot.** A chatbot answers. An agent plans and takes action.
- ★ **Embedding model vs. text model.** Two models, two jobs.
- ★ **Responsible AI vs. security.** Behavior controls versus access and data controls.
- ★ **Real-time vs. async vs. batch.** Waiting user, big or slow job, or offline bulk.
- ★ **Supervised vs. unsupervised.** Labels or no labels.
- ★ **Vector DB vs. traditional DB.** Similarity versus exact match.
- ★ **Bigger isn't always better.** Cost, latency, and fit can beat raw size.
- ★ **RAG lowers risk, not to zero.** Retrieval quality still decides the answer.
- ★ **Human vs. automated evaluation.** Nuance versus speed and scale.
- ★ **Prompt vs. context engineering.** How you ask versus what the model sees.
- ★ **On-demand vs. Provisioned.** Pay per token, or reserve capacity for steady load.
- ★ **Foundation vs. fine-tuned model.** General-purpose versus adapted to your data.
- ★ **Lineage vs. Model Card.** Path of the data, versus description of the model.
- MemoryDB.** Dropped from the current list. Don't pick it for vector search.
- Amazon Q.** In v1.1 scope but recognition-level. Don't pick it when the clue says multiple FMs through one API (Bedrock).
- ROUGE vs. BLEU.** Summaries versus translation.
- Distillation vs. fine-tuning.** Small model mimics a big one, versus adapting one model.
- Pre-training variants.** Scratch, labeled tuning, or continued on unlabeled data.
- Shared responsibility.** Managed doesn't mean AWS owns your data and access.

AI/ML terminology quick reference

Model: trained artifact you call
Algorithm: the learning recipe
Inference: using a trained model
Token: text chunk, billed unit
Context window: max tokens per request
Embedding: meaning as a vector
Vector DB: nearest-vector search
Chunking: splitting docs for RAG

FM: broad pre-trained model
LLM: FM for language
RAG: retrieve, then generate
Fine-tuning: retrain on labeled data
Distillation: small mimics big
Temperature: output randomness
Few-shot: examples in prompt
Hallucination: confident, false output

Overfitting: memorized training data
Recall: real positives caught
Precision: flagged items truly positive
Agent: FM + tools + memory + loop
MCP: standard for tool access
Guardrails: content controls
Model Card: model documentation
Lineage: data's path to the model

Responsible AI checklist

- ☐ **Fair?** tested across groups
- ☐ **Safe?** guardrails and human review
- ☐ **Accurate?** grounded and evaluated
- ☐ **Transparent?** sources and limits documented
- ☐ **Explainable?** decisions can be justified
- ☐ **Private?** sensitive data protected
- ☐ **Robust?** holds up on messy or hostile input
- ☐ **Inclusive?** data reflects real users

Security & governance quick reference

IAM	Who can do what
KMS	Encryption keys
Macie	Sensitive data in S3
PrivateLink	Private connectivity
CloudTrail / Config	API history / config rules
Artifact / Inspector	Compliance reports / vulnerabilities
AgentCore Identity / Policy	Who the agent is / what it may do

48-hour revision checklist

48 to 36 hours out

- ☐ Take one timed practice exam
- ☐ Tag every miss by domain
- ☐ Reread any domain under 75%

36 to 24 hours out

- ☐ Redraw the RAG diagram from memory
- ☐ Recite the customization table
- ☐ Cover the answers on the clue pages

24 to 12 hours out

- ☐ Skim the ★ traps
- ☐ Review the service map
- ☐ Light practice only. No cramming

Exam day

- ☐ Sleep and eat
- ☐ ID and test logistics ready
- ☐ Read the final 15-minute review

Final 15-minute review: say these out loud

1. Bedrock uses FMs. SageMaker AI builds models.
2. Stale or private facts → RAG. Behavior and style → fine-tuning.
3. Prompt first, then RAG, then fine-tune.
4. Low temperature = predictable. High = varied.
5. Embeddings compare meaning. Tokens get counted.
6. Guardrails control content. IAM controls access.
7. Macie finds sensitive data. KMS holds keys.
8. Identity = who. Policy = what's allowed.

9. Classification = category. Regression = number.
10. Recall for costly misses. Precision for costly false alarms.
11. Hallucination ≠ bias. Transparency ≠ explainability.
12. Test on your own data before choosing a model.

ON EXAM DAY Read the last line of the question first. Cross out the clearly wrong answers. Match the *exact* requirement: cheapest, fastest, least effort, most secure. Flag anything uncertain, finish the exam, then come back. Change an answer only if you can name why.

YOU'VE GOT THIS You don't need to know everything about AI. You need to recognize the pattern in the question and match it to the right concept or AWS service. This sheet gives you the patterns. Go pass it.