MockCert

Study guide | Amazon Web Services | AIP-C01

AWS Certified Generative AI Developer - Professional

Notes for every domain of the AWS Certified Generative AI Developer - Professional exam, organized by the official exam guide's domains and weights.

By MockCert about 79 min read Updated November 14, 2026

CONTENTS

DOMAIN 1 · 31%
Foundation Model Integration, Data Management, and Compliance 2

DOMAIN 2 · 26%
Implementation and Integration 18

DOMAIN 3 · 20%
AI Safety, Security, and Governance 31

DOMAIN 4 · 12%
Operational Efficiency and Optimization for GenAI Applications 42

DOMAIN 5 · 11%
Testing, Validation, and Troubleshooting 50

MockCert is an independent certification preparation platform. It is not affiliated with, endorsed by or operated by Amazon Web Services. Exam domains and weights follow the official exam guide. These notes were last updated on November 14, 2026; the notes at mockcert.com are kept up to date as the exam changes.

Domain 1 | 31% of the exam | 23 min read

Foundation Model Integration, Data Management, and Compliance

Overview

This domain is about 31% of the exam, roughly 23 of its 75 questions.

Every point below comes from AWS's official documentation; each section lists its sources.

Analyze requirements and design GenAI solutions.

Start here

THINK FIRST

How would you design, prove and roll out a GenAI solution on AWS before trusting it with real production traffic?

Why it matters: Questions describe a business need and ask which Bedrock or SageMaker AI building block, test or rollout choice fits it.

Start small: an agent can be created with only a name, and a flow needs one input node and at least one output node. Compare a new variant in a shadow test that returns no responses. Then shift traffic while CloudWatch alarms watch the green fleet during a baking period.

Key terms

Term	Meaning
Shadow variant	A model variant that gets a copy of some production requests but sends no responses back.
Baking period	The set time CloudWatch alarms watch the green fleet before SageMaker AI finishes a rollout.
Collector node	A flow node placed after an iterator node that returns the outputs as an array.
Embedding model type	A managed knowledge base setting, `MANAGED` or `CUSTOM`; it is `MANAGED` when omitted.

Start an agent small

An agent can be created with only a name. Prepared with no action groups or knowledge bases, it answers from the FM, its instructions and the base prompt templates. For cross-Region inference, put an inference

profile ID in the `foundationModel` field.

EXAM TRAP

An agent does not keep a conversation forever. By default it drops the history after **30** minutes with no reply from the user.

Flow structure rules

A flow has exactly one input node and at least one output node. Console-created flows start with a single prompt node. In a condition node, every condition, the default one included, must link to a downstream node. After an iterator node, add a collector node to gather the results.

EXAM TRAP

Deleting nodes does not shrink an automatically created service role. It gains permissions as you add nodes, but keeps them when nodes are removed.

Managed or custom embeddings

The embedding model type is fixed when the knowledge base is created. Moving from managed to custom embedding, or back, takes a new knowledge base. So decide during design, not after launch.

Managed and custom embedding in a managed knowledge base

Feature	MANAGED	CUSTOM
When it applies	Default when `embeddingModelType` is omitted	Only when you set it explicitly
Model fields	Leave out `embeddingModelArn` and `embeddingModelConfiguration`	Both `embeddingModelArn` and `embeddingModelConfiguration` required
Model setup	Service-managed; no selection or configuration	**1024** dimensions and float32 data type
Managed reranker	Available	Not available

EXAM TRAP

A managed knowledge base is queried with `managedSearchConfiguration` in the Retrieve API. `vectorSearchConfiguration` is meant for custom knowledge bases only.

Prove it with a shadow test

A shadow test compares a shadow variant with the production variant. The shadow variant gets a replicated share of requests but returns nothing to callers. Each endpoint allows at most **one** shadow variant. A test runs **7** days by default and **30** days at most.

EXAM TRAP

Shadow tests are not for every endpoint. Serverless inference, asynchronous inference and multi-model endpoints cause validation errors.

Roll out with guardrails

SageMaker AI updates models with blue/green or rolling deployments. Deployment guardrails cover only asynchronous and real-time inference endpoints. Canary shifting moves traffic in **two** steps; linear shifting uses n evenly spaced steps.

All at once and canary traffic shifting compared

Feature	All at once	Canary
First shift	All traffic moves to the green fleet	One small portion moves to the green fleet
During the baking period	CloudWatch alarms watch the green fleet	The canary on the green fleet is monitored
Afterwards	Blue fleet terminated if no alarm trips	Rest of the traffic shifts, then the blue fleet ends

Reusable starting points

JumpStart solutions are one-click, end-to-end starting points. They target common machine learning use cases. They are available only in Studio Classic, so check where your team works first.

Test prompts honestly

Keep a hold-out test set for the final best prompt only. Checking only at the end shows whether prompt changes generalize to unseen data. For simple tasks such as text classification, **3** to **5** few-shot examples can be enough.

Key takeaways

- A flow needs exactly **one** input node, and every condition in a condition node needs a downstream link.
- Choose managed or custom embedding up front; changing it means a new knowledge base.
- A shadow variant sees replicated traffic but answers no one, and an endpoint gets only **one**.
- Canary and all at once shifting both end the blue fleet only after a clean baking period.

Sources: Create and configure agent manually - Amazon Bedrock (opens in a new tab) · Design a flow in Amazon Bedrock - Amazon Bedrock (opens in a new tab) · Create a shadow test - Amazon SageMaker AI (opens in a new tab) · Deployment guardrails for updating models in production - Amazon SageMaker AI (opens in a new tab) · Create a managed knowledge base - Amazon Bedrock (opens in a new tab) · Create and design a flow in Amazon Bedrock - Amazon Bedrock (opens in a new tab) · End-to-end JumpStart solution templates - Amazon SageMaker AI (opens in a new tab) · Design a prompt - Amazon Bedrock (opens in a new tab)

Select and configure FMs.

Start here

THINK FIRST

How do you pick, run and later replace a foundation model on Bedrock without your application breaking along the way?

Why it matters: Questions describe a model's lifecycle stage, a hosting need or an outage risk and ask which configuration or action fits.

Every Bedrock model is Active, Legacy or End-of-Life (EOL). Legacy blocks new fine-tuning jobs, and you must migrate your code yourself before the EOL date. Plan for change: route across more Regions when your workload does not need one Region.

Key terms

Term	Meaning
Legacy state	The stage before EOL: new customers can't adopt the model, and new fine-tuning jobs are blocked.
Provisioned Throughput	Purchased dedicated compute with guaranteed throughput for steady performance and lower latency.
modelLifecycle	The response field of GetFoundationModel and ListFoundationModels that shows a model's state.
Public extended access	A late part of the Legacy period where active users keep the model at higher provider pricing.
Inference profile	A resource with its own ARN that you reach a model through; cross-Region ones, such as US or EU, route requests across more Regions.

Three lifecycle states

Every Bedrock model is Active, Legacy or End-of-Life (EOL). A model stays on Bedrock for at least **12** months after launch before its EOL date. States can differ by Region, so call GetFoundationModel or ListFoundationModels in the Region you use.

EXAM TRAP

Dates a provider such as Anthropic or Cohere publishes do not govern Bedrock. Only the dates on the Bedrock model card apply.

What Legacy blocks

Legacy stops new customers and new fine-tuning jobs. Most models stay in Legacy for **6** months; the other Legacy period is **45** days. Existing customers may lose access after **15** days without use.

Actions on a base model before and during Legacy

Action	Active	Legacy
New customers adopt the model	Allowed	Not allowed
New Provisioned Throughput	Allowed	Not allowed
New fine-tuning jobs	Allowed	Not allowed
New on-demand deployment of a model customized earlier	Allowed	Still allowed

Reaching End-of-Life

After EOL, the model leaves every Region and its requests fail. Only a private arrangement with the provider keeps access. Migration never happens on its own: update your application code to the newer model before the EOL date.

EXAM TRAP

Do not count on an inference profile to move you to a successor. A successor may simply be missing from that same inference profile.

Extended access costs more

Late in Legacy, active users can stay on at a higher price. This rule covers models whose EOL date falls after **February 1, 2026**. After at least **3** months in Legacy, public extended access begins, and the provider sets the higher pricing.

Hosting a custom model

On-demand bills per use; Provisioned Throughput buys dedicated capacity. Invoke an on-demand custom model with the ARN of its custom model deployment. Customized Nova models are registered in SageMaker AI Model Registry and deploy as Bedrock imported models. Customized open weight models go to SageMaker AI Inference endpoints.

On-demand and Provisioned Throughput for custom models

Feature	On-demand	Provisioned Throughput
Billing	Pay only for what you use	Purchased capacity
Compute	No provisioned resources to set up	Dedicated compute capacity
Main benefit	No provisioned compute to set up	Guaranteed throughput, consistent performance, lower latency

Running a customization job

Without `customizationType`**, the job defaults to** `FINE_TUNING`**.** The base model can be a foundation model or a model you already fine-tuned or distilled. Training and validation loss for each epoch land in separate files in your output location.

EXAM TRAP

Adding a VPC configuration has a side effect. The console can no longer create a new service role for that customization job.

Routing for resilience

In-Region inference keeps processing in one Region for data residency. Workloads without that need can use global or geographic profiles, such as US or EU, which spread requests across more Regions. The exam guide pairs resilience with Step Functions circuit breakers and graceful degradation. It pairs model switching without code changes with Lambda, API Gateway and AppConfig.

Evaluating candidate models

An automatic evaluation job can test an inference profile. In the API, put the profile's ARN in `modelIdentifier`. A human-based job compares up to **two** inference sources, which need not be the same type. Outside the console it also needs a SageMaker AI flow definition ARN.

Key takeaways

- Legacy blocks new customers, new Provisioned Throughput and new fine-tuning jobs.
- After EOL, requests fail in all Regions, and you must migrate your code yourself beforehand.
- A model stays at least **12** months after launch, and only the Bedrock model card dates count.
- Choose in-Region inference for data residency and cross-Region profiles for wider routing.

Sources: Model lifecycle - Amazon Bedrock (opens in a new tab) · Model lifecycle (Legacy) - Amazon Bedrock (opens in a new tab) · Inference profiles with limited model selection - Amazon Bedrock (opens in a new tab) · Set up inference for a custom model - Amazon Bedrock (opens in a new tab) · Starting an automatic model evaluation job in Amazon Bedrock - Amazon Bedrock (opens in a new tab) · Deploying customized models - Amazon SageMaker AI (opens in a new tab) · Submit a model customization job for fine-tuning - Amazon Bedrock (opens in a new tab) · Exam Guide (AIP-C01) AWS Certified Generative AI Developer - Professional (opens in a new tab)

Implement data validation and processing pipelines for FM consumption.

Start here

THINK FIRST

What has to happen to text, audio, images and tables before a foundation model can use them reliably?

Why it matters: Questions describe messy or mixed input data and ask which validation, processing or formatting step makes it fit for the model.

Good answers start with good input. Validate data and watch it for drift. Process each data type with the right service, then send it in the format the model expects.

Key terms

Term	Meaning
Inference pipeline	A SageMaker AI model built from a linear chain of containers that process each request in turn.
Data capture	Stores inference input and output from a real-time endpoint or batch transform job in Amazon S3.
Baseline	Schema constraints and statistics for each feature, computed with Deequ on Apache Spark.
Model invocation type	The batch inference input format choice: InvokeModel by default, or Converse.
SageMaker Clarify job	A SageMaker processing job that reads an input dataset and sends requests to a model endpoint, set up in an analysis configuration.

Validate and enhance input

Check data quality before any FM sees it. For validation, the exam guide names Lambda functions, CloudWatch metrics, AWS Glue Data Quality and SageMaker Data Wrangler. To lift input quality, reformat text with Bedrock, extract entities with Comprehend and normalize data with Lambda.

Watching for data drift

Model Monitor applies rules to spot data drift and alert you. Data capture saves endpoint or batch transform traffic to S3. A baseline job computes schema constraints and statistics for each feature with Deequ.

EXAM TRAP

Model Monitor computes metrics and statistics on tabular data only, so it does not profile raw text or images. It is also closed to new customers.

Preprocess in an inference pipeline

An inference pipeline chains 2 to 15 containers in a line. Each container's response becomes the next one's request, and the client gets the final response. All containers run on every instance, so feature processing stays fast. Spark ML Serving and scikit-learn containers can reuse your training transformers.

EXAM TRAP

You cannot edit a deployed pipeline model in place, because it is immutable. Deploy a new pipeline with the UpdateEndpoint operation instead.

Processing audio data

Bedrock Data Automation labels, splits and screens audio. For mixed data types, the exam guide also lists Bedrock multimodal models, SageMaker Processing and Transcribe.

Bedrock Data Automation audio features

Feature	What you get	Limit or default
Speaker labels	`spk_` plus a unique number	Up to **30** speakers
Channel labels	First channel is `ch_0`	Up to **two** channels
Topic summary	Timestamped topics, each summarized	Not enabled by default
Content moderation	Voice-based toxic content classified	**Seven** categories

Batch inference input

Batch inference reads every JSONL and content file at the S3 location. The location can be a folder or a single JSONL file. The `timeoutDurationInHours` field defaults to **24** hours and accepts **24** to **168**.

> **EXAM TRAP**
>
> The default invocation type, InvokeModel, expects model-specific request formats. If your records use the Converse API format, choose Converse instead.

Addressing the model correctly

With the OpenAI SDK, point requests at the Bedrock endpoint. Do not use the OpenAI endpoint. Closed-weight OpenAI GPT models also need a system-defined inference profile in the request, not the foundation model ID.

Formats in a Clarify job

Dataset, request and response formats are set separately. They do not have to match. A Parquet dataset can pair with CSV requests and JSON Lines responses.

SageMaker Clarify format parameters

Parameter	Sets the format of	Image data note
`dataset_type`	The input dataset	Set to `application/x-image`
`content_type`	Requests to the endpoint	Decides the file format; `image/jpeg` if omitted
`accept_type`	Responses from the endpoint	Can differ from the request format

Testing an endpoint by hand

Check requests and responses on a real-time endpoint first. If the container supports batch requests, start with one record, then try two or more. With AWS CLI v2, add `--cli-binary-format`

`raw-in-base64-out`; with `v1`, leave it blank.

Key takeaways

- An inference pipeline runs **2** to **15** co-located containers and costs nothing beyond its instances.
- Bedrock batch inference defaults to InvokeModel and a **24**-hour timeout.
- Model Monitor uses rules to detect data drift, and it computes statistics on tabular data only.
- In Clarify, `dataset_type`, `content_type` and `accept_type` can each use a different format.

Sources: Exam Guide (AIP-C01) AWS Certified Generative AI Developer - Professional (opens in a new tab) · Data quality - Amazon SageMaker AI (opens in a new tab) · Create a batch inference job - Amazon Bedrock (opens in a new tab) · Inference pipelines in Amazon SageMaker AI - Amazon SageMaker AI (opens in a new tab) · Data Format Compatibility Guide - Amazon SageMaker AI (opens in a new tab) · Image data requirements - Amazon SageMaker AI (opens in a new tab) · Audio - Amazon Bedrock (opens in a new tab) · Responses API - Amazon Bedrock (opens in a new tab)

Design and implement vector store solutions.

Start here

THINK FIRST

How do metadata, search settings and tool discovery decide whether a model finds the right context quickly?

Why it matters: Questions describe a retrieval or search need and ask which metadata, search setting or integration choice delivers it.

Retrieval is only as precise as what you can search on. Rich metadata narrows Feature Store searches, semantic search picks the right gateway tool, and a lean agent design cuts LLM calls and latency.

Key terms

Term	Meaning
Feature parameter	A searchable key-value pair attached to a Feature Store feature; up to **25** per feature.
Discoverable feature group	A feature group another account shared and made discoverable, so your searches can find it.
Gateway semantic search	AgentCore gateway search that finds the best tools with a natural language query.
TSTALIASID	The agent alias used in an InvokeAgent request to test a prepared Bedrock agent.

Metadata makes features findable

Give each feature a description and searchable parameters. A description holds up to **255** characters, and a feature takes up to **25** parameters. First confirm with DescribeFeatureGroup that the group status is Created. Update with the console or UpdateFeatureMetadata, then check with DescribeFeatureMetadata.

EXAM TRAP

A console search on a parameter needs both its key and its value. The key alone is not enough.

Searching features

You can search all features without choosing a feature group first. Use the console or the Search operation. When a Search expression has several filters, they combine with And unless you set the operator yourself.

EXAM TRAP

The feature group name field, FeatureGroupName, is not searchable in the console, so search on other fields there.

Shared feature groups

Search finds shared groups only when they are discoverable to you. Otherwise a group must be in your own Region and account. On shared groups, tags are not searchable, and editing needs permission from the owner account.

Your own and shared feature groups in Feature Store search

Feature	Your own groups	Shared, discoverable groups
Found by search	Yes, in your Region and account	Yes, once shared and made discoverable
Tags searchable	Yes	No
Edit metadata or definitions	Yes	Only with access permission from the owner account

Semantic tool search

Gateway semantic search helps when you have many tools. Turn it on in `create_gateway` by setting `searchType` to `SEMANTIC` under `protocolConfiguration.mcp`. Then call the `x_amz_bedrock_agentcore_search` tool through a POST `tools/call` request to the gateway's MCP endpoint.

EXAM TRAP

Semantic search is a create-time setting, not an option of the search tool. If you enabled it when creating the gateway, you can call `x_amz_bedrock_agentcore_search`.

Gateway speed and versions

Gateway latency depends mostly on the tools behind it. Keep Lambda functions in the gateway's Region and give latency-sensitive ones provisioned concurrency. The gateway also accepts only the MCP protocol versions listed in `supportedVersions`.

Fewer LLM calls per request

An agent with one knowledge base and no active action groups can skip LLM calls. When those conditions hold, the service lowers the LLM calls each request needs. Apply changes with PrepareAgent, then test with InvokeAgent and the TSTALIASID alias.

> **EXAM TRAP**
>
> Overriding the default orchestration prompt template sends the agent back to the standard flow, which may add LLM calls and latency. Reverting deletes your custom template.

Network and access setup

Studio and notebook instances allow direct internet access by default. Launch them in a VPC to restrict which traffic reaches the internet. Feature Store example notebooks need the AmazonSageMakerFeatureStoreAccess policy on the execution role. Sharing with the AWS RAM Python SDK needs the AWS RAM full access policy.

Key takeaways

- A feature takes a description of up to **255** characters and up to **25** searchable parameters.
- Tags are searchable only on your own feature groups, not on discoverable ones.
- Enable gateway semantic search at creation, then query tools in natural language.
- Overriding the default orchestration template may add LLM calls and latency.

Sources: Adding searchable metadata to your features - Amazon SageMaker AI (opens in a new tab) · Search for tools in your AgentCore gateway with a natural language query - Amazon Bedrock AgentCore (opens in a new tab) · Optimize performance for Amazon Bedrock agents using a single knowledge base - Amazon Bedrock (opens in a new tab) · Find feature groups in your Feature Store - Amazon SageMaker AI (opens in a new tab) · Find features in your feature groups - Amazon SageMaker AI (opens in a new tab) · Performance optimization - Amazon Bedrock AgentCore (opens in a new tab) · Connect to Amazon SageMaker AI resources from within a VPC - Amazon SageMaker AI (opens in a new tab) · Amazon SageMaker Feature Store resources - Amazon SageMaker AI (opens in a new tab)

Design retrieval mechanisms for FM augmentation.

Start here

THINK FIRST

When a model needs outside context, how do you decide what to retrieve, from where, and how much of it?

Why it matters: Questions describe a retrieval goal and ask which chunking, search, query or Web Search setting improves the results.

Retrieval quality is set before the model answers. Split documents sensibly, search with keywords and vectors, then shape queries. When fresh facts are needed, Web Search lets the model look things up within the domain filters you set.

Key terms

Term	Meaning
Hybrid search	A search architecture that combines keyword matching and vectors to improve relevance.
search_context_size	The Web Search setting that sets how many observations the model may use; medium by default.
external_web_access	The Web Search setting that decides whether retrieval may leave the AWS boundary; true by default.
MCP	Model Context Protocol: the standard whose clients run vector queries and whose endpoint a gateway exposes.
x_amz_bedrock_agentcore_search	The AgentCore gateway tool that finds gateway tools from a natural language query.

Segmenting and embedding

How you chunk documents shapes retrieval performance. The exam guide names Bedrock chunking capabilities, Lambda for fixed-size chunking and custom hierarchical chunking that follows content structure. For embeddings, pick Titan models by dimensionality and domain fit, and batch-generate vectors with Lambda.

Search and query handling

Hybrid search mixes keywords with vectors for better relevance. Bedrock reranker models and OpenSearch Service vector search also appear in the exam guide. For queries, it pairs expansion with Bedrock, decomposition with Lambda and transformation with Step Functions.

Turning on Web Search

Put a `web_search` tool in the request's `tools` array. The model then decides whether a request needs current information. If the first results fall short, it can rewrite its query and search again in the same turn.

EXAM TRAP

Web Search runs server-side, so calls to the Responses API on the bedrock-runtime endpoint cannot use it.

Sizing the search context

`search_context_size` **sets an observation budget, medium by default.** Larger budgets add context but can use more input tokens.

Two search_context_size settings compared

Setting	Observation budget	Best for
medium (default)	Up to **11**	Balancing coverage and input-token use
high	Up to **25**	Complex or multi-hop questions

EXAM TRAP

`search_context_size` does not cap how many Search calls the model makes. It only budgets the observations.

Filtering web domains

The `filters` **parameter takes up to 100 allowed and 100 blocked domains.** Leave the protocol prefix off each domain. Every entry also covers its subdomains, so `aws.amazon.com` includes `docs.aws.amazon.com`.

Staying inside AWS

Set `external_web_access` **to false to keep retrieval within AWS.** Cached page content is still allowed. The default is true, matching the OpenAI Responses API so existing calls need no change.

Tools through a gateway

Gateway semantic search finds the right tool among many. Enable it at creation with `searchType` set to `SEMANTIC`. Then send a POST `tools/call` request to the gateway's MCP endpoint. Clear parameter descriptions and required fields help agents call tools correctly.

EXAM TRAP

A gateway rejects MCP protocol versions missing from `supportedVersions` in `protocolConfiguration.mcp`.

Lean agent retrieval

One knowledge base and no active action groups can mean fewer LLM calls. The agent must also keep the default orchestration template and not ask the user for more information. Confirm with ListAgentKnowledgeBases that one summary returns. Apply changes with PrepareAgent and test with the

TSTALIASID alias.

EXAM TRAP

Setting `promptCreationMode` to DEFAULT through UpdateAgent deletes your custom template. Save it from `basePromptTemplate` first.

Key takeaways

- Enable Web Search with a `web_search` tool; the model decides when to use it.
- Medium allows up to **11** observations and high up to **25**, with no cap on Search calls.
- Domain filters match subdomains and go without a URL prefix.
- For lower gateway latency, keep Lambda functions in the gateway's Region with provisioned concurrency.
- The exam guide lists function calling and MCP clients as consistent ways to reach vector search.

Sources: Exam Guide (AIP-C01) AWS Certified Generative AI Developer - Professional (opens in a new tab) · Web Search - Amazon Bedrock (opens in a new tab) · Search for tools in your AgentCore gateway with a natural language query - Amazon Bedrock AgentCore (opens in a new tab) · Performance optimization - Amazon Bedrock AgentCore (opens in a new tab) · Optimize performance for Amazon Bedrock agents using a single knowledge base - Amazon Bedrock (opens in a new tab)

Implement prompt engineering strategies and governance for FM interactions.

Start here

THINK FIRST

How do you write, manage and keep improving prompts so a model behaves the same way every time?

Why it matters: Questions describe a prompt that misbehaves or needs control and ask which technique, setting or Bedrock feature fixes it.

A prompt is a managed asset, not a one-off string. Structure it carefully, give it examples and context, store it with variables and versions, and prove each change on unseen hold-out data.

Key terms

Term	Meaning
Few-shot prompting	Giving the model paired example inputs and desired outputs; also called in-context learning.
Prompt variable	A placeholder in double curly braces that is filled in when the prompt is called.
Prompt variant	An alternative version of a prompt compared side by side during testing.
Hold-out test set	Sample data kept unseen and used only to verify the final best prompt.

Term	Meaning
Payload referencing	Prompt compression for multi-agent collaboration that cuts the primary agent's output tokens.

Instructions that steer output

Put the task or question at the end of the prompt. For classification, list the answer choices at the end too. Give a default output for when the model is unsure, so low-confidence answers do not sound right. Temperature runs from **0** to **1**; lower values give more deterministic responses.

Few-shot examples

Examples help the model match the output you expect. For text classification, **3** to **5** examples can suffice; harder tasks, such as answering without context, need more. With Claude models, wrap examples in example tags. With Titan, asking it to think step-by-step can help on complex tasks.

Keeping conversation context

Models called through the API do not remember earlier requests. Include the earlier exchange in the current prompt. For models that support the Converse API, a managed prompt can hold a system prompt plus past user and assistant messages.

> **EXAM TRAP**
> An agent session does not last forever. Reusing the session ID keeps it going only while `idleSessionTTLInSeconds` has not expired.

Building managed prompts

Managed prompts use variables such as `{{variable}}` to fit many use cases. CreatePrompt returns an ID and ARN and starts a DRAFT version. Every prompt supports `maxTokens`, `stopSequences`, `temperature` and `topP`. You connect a prompt to your application through Bedrock Flows.

> **EXAM TRAP**
> Prompt caching needs the CHAT template type, and CHAT works only with models that support the Converse API. A TEXT prompt cannot use caching.

Versions and variants

Save versions while you iterate; numbering starts at 1. CreatePromptVersion returns an ID and ARN for each version. When testing, compare up to **three** variants and keep the one whose outputs suit your use case best.

Optimizing a prompt

OptimizePrompt tunes a prompt for the model named in `targetModelId`. Simple optimization works best for prompts of about **1k** tokens or less. To cut hallucinations, refine the prompt, add relevant data with RAG or try another model.

EXAM TRAP

Simple prompt optimization uses no evaluation data and cannot target several models at once.

Testing for real improvement

Divide sample data into development and hold-out test subsets. Try prompt changes on the development set and watch how responses shift. Touch the test set only once, to confirm the final best prompt generalizes.

Agent prompt controls

Turn on the trace to see an agent's step-by-step process. In advanced prompts, pre-processing is disabled by default for all supported models. One parser Lambda function can serve every base template, with invocation set per step. Payload referencing is on by default for the primary agent.

EXAM TRAP

Disabling the orchestration step skips its template entirely. The agent passes raw user input straight to the foundation model.

Key takeaways

- Place the task and any answer choices at the end of the prompt.
- API calls are stateless: include earlier turns in the prompt to keep context.
- Prompt caching needs the CHAT template, which needs a Converse API model.
- Use the hold-out test set only to verify the final prompt.

Sources: Design a prompt - Amazon Bedrock (opens in a new tab) · Prompt engineering concepts - Amazon Bedrock (opens in a new tab) · Create a prompt using Prompt management - Amazon Bedrock (opens in a new tab) · Construct and store reusable prompts with Prompt management in Amazon Bedrock - Amazon Bedrock (opens in a new tab) · Enhance agent's accuracy using advanced prompt templates in Amazon Bedrock - Amazon Bedrock (opens in a new tab) · Create a version of a prompt in Prompt management - Amazon Bedrock (opens in a new tab) · Optimize a prompt - Amazon Bedrock (opens in a new tab) · Advanced prompt templates - Amazon Bedrock (opens in a new tab)

Domain 2 | 26% of the exam | 19 min read

Implementation and Integration

Overview

This domain is about 26% of the exam, roughly 20 of its 75 questions.

Every point below comes from AWS's official documentation; each section lists its sources.

Implement agentic AI solutions and tool integrations.

Start here

THINK FIRST

What does an agent need beyond a model to remember, follow rules, use tools and bring people in?

Why it matters: Questions describe an agent's memory, safety, tool or review need and ask which AgentCore or Bedrock feature or setting meets it.

An agent pairs a model with supporting services. AgentCore Memory stores raw events, and Automated Reasoning translates a source document into formal logic rules. Semantic search on a gateway helps agents find tools, and human workers can review model responses.

Key terms

Term	Meaning
Memory strategy	A long-term memory setting such as SEMANTIC, SUMMARIZATION, USER_PREFERENCE or EPISODIC.
Short-term memory	The raw events AgentCore Memory stores; all it keeps when no strategy is defined.
Automated Reasoning policy	Formal logic rules plus a schema of variables and types, built from a source document.
Curated query	A natural language question paired with its SQL, used as an example for SQL generation.

Short-term and long-term memory

With no strategy, AgentCore Memory stores only short-term raw events. Add a long-term strategy, such as SEMANTIC or EPISODIC, to keep more than raw events. Each kind of memory has its own retention rule.

Retention for raw events and long-term records

Feature	Raw events	Long-term records
Retention range	**3** to **365** days in the CLI wizard	`recordExpiryDurationDays`, at least **31** days
If you set nothing	**30** days, the CLI wizard default	Records do not expire
Where you set it	Entered in the CLI Memory wizard	In each strategy's `recordExpirySpecification`

EXAM TRAP

Raising `eventExpiryDuration` later does not extend stored events. It applies at write time, so only new events change, and expired events cannot be recovered.

Securing and watching memory

Front Memory with a gateway to add OAuth and fine-grained access control. That enforces per-user isolation at the infrastructure layer without handing AWS credentials to end users. Data is encrypted with a service managed key even without a customer managed key. Metrics go to the Bedrock-AgentCore CloudWatch namespace.

Debugging memory ingestion

Ingestion logs land in a log group in your own account. This happens when log delivery is enabled. Use those application logs to find errors in the asynchronous ingestion process.

Rules the model must follow

Automated Reasoning turns a source document into formal logic rules. Source documents are limited to **5** MB and **50,000** characters. The fidelity report helps you review rules with the experts who wrote the document. A policy allows **2** build workflows, only **1** in progress at once.

EXAM TRAP

Unexpected IMPOSSIBLE results can come from conflicting rules or from bare assertions. Bare assertions create axioms that make some conditions logically impossible.

Finding the right tool

Gateway semantic search helps agents pick the right tools. Turn it on with `searchType` set to `SEMANTIC` in the gateway's MCP protocol configuration. Remember that gateway latency comes mostly from the tools behind it.

Leaner agent flows

Agents with one knowledge base can use latency-optimized flows. When the optimization conditions are met, the service cuts the LLM calls needed for each request.

EXAM TRAP

Overriding the default orchestration prompt template moves the agent to the standard flow, which may add LLM calls and latency.

Querying structured data

A query engine, such as Redshift, turns questions into SQL. In CreateKnowledgeBase, set the type to SQL and include `sqlKnowledgeBaseConfiguration`. Curated queries give the SQL generator worked examples to follow.

EXAM TRAP

Inclusions are not additive hints. Once you specify inclusions, every other table and column is ignored.

Bringing in human experts

A human evaluation job compares up to two inference sources. One can be a Bedrock model and the other your own response data. A prompt leaves circulation once enough workers review it. Outside the console, the flow definition's `AwsManagedHumanLoopRequestSource` must be `AWS/Bedrock/Evaluation`.

Key takeaways

- No memory strategy means only short-term raw events are stored.
- Long-term records never expire unless `recordExpiryDurationDays` is set, at **31** days or more.
- Conflicting rules make an Automated Reasoning policy return IMPOSSIBLE.
- Automated Reasoning policies use a service-owned KMS key unless you choose otherwise.

Sources: Create an AgentCore Memory - Amazon Bedrock AgentCore (opens in a new tab) · Create your Automated Reasoning policy - Amazon Bedrock (opens in a new tab) · Performance optimization - Amazon Bedrock AgentCore (opens in a new tab) · Create a human-based model evaluation job - Amazon Bedrock (opens in a new tab) · Create a knowledge base by connecting to a structured data store - Amazon Bedrock (opens in a new tab) · Advanced features and topics for Amazon Bedrock AgentCore Memory - Amazon Bedrock AgentCore (opens in a new tab) · Observability - Amazon Bedrock AgentCore (opens in a new tab) · Optimize performance for Amazon Bedrock agents using a single knowledge base - Amazon Bedrock (opens in a new tab)

Implement model deployment strategies.

Start here

THINK FIRST

How do you deploy a large language model so it is fast enough, scales with demand and does not waste expensive hardware?

Why it matters: Questions describe an FM workload's latency, scale or cost need and ask which deployment

target, serving setting or scaling option fits.

LLM deployments must balance speed, scale and hardware cost. Good deployments choose the right host, share hardware, reuse cached work and scale on the right signals, down to zero when idle.

Key terms

Term	Meaning
Inference component	A SageMaker AI hosting unit whose copies can scale down to zero to free room for others.
KV cache	Stored attention keys and values, kept to optimize inference latency.
Multi-Instance GPU (MIG)	GPU partitioning that runs several inference workloads on a single GPU.
Dynamic batching	A DJL Serving feature that groups requests to increase throughput.

Choosing where it runs

Customized Nova models deploy to Bedrock; open weight models to SageMaker AI. Keep the S3 bucket with model artifacts in the same Region as the model. Inference Recommender suggests instance types, and popular JumpStart LLMs come pre-benchmarked for cost or performance.

Endpoint models on SageMaker AI

A Single Model Endpoint hosts one model per variant, without components. With inference components, any copy can scale to zero, freeing room for another. In the console, a component needs at least **128** MB of memory. Timeout settings cap model download and container startup time.

Serving with DJL Serving

DJL Serving runs models on CPUs, GPUs and AWS Inferentia. In Python mode, set only `option.model_id` in `serving.properties`. By default, it sizes the worker count from the available hardware. Set `required_memory_mb` and `reserved_memory_mb` for each model to control memory.

Token limits in JumpStart

Max total tokens is input tokens plus generated output. You can also set Max input token length for each request and Max concurrent requests the model handles at once.

> **EXAM TRAP**
>
> Max total tokens is not an output limit alone. A long prompt uses up part of the same budget.

Deploying on HyperPod

HyperPod deploys through kubectl, the Python SDK, Studio or its CLI. To cut cold starts, cache weights on host-local NVMe and pre-pull the container image with `modelCacheConfig`. A prioritized list of instance types gives automatic fallback when capacity runs short. Observability tracks time-to-first-token, latency and GPU use.

KV cache and routing

HyperPod's KV cache has an L1 tier in CPU memory and an L2 tier in Redis. L1 gives low-latency local reuse; L2 shares cache across nodes. Routing then steers requests toward cached work.

HyperPod cache-aware routing strategies

Strategy	Sends a request to	Based on
prefixaware	The instance that handled the same prompt prefix	Prompt prefix match
kvaware	The instance whose KV cache hit rate is highest	KV cache hit rate

EXAM TRAP

KV cache data is stored unencrypted at rest. With managed tiered storage as L2, deployments in one cluster share cache with no isolation.

Autoscaling with KEDA

By default, autoScalingSpec keeps between 1 and 5 replicas. CPU and memory triggers can reach zero only alongside another scaler, such as SQS or Prometheus.

Key autoScalingSpec settings

Setting	Default	Note
minReplicaCount	**1**	Must be **0** or more
maxReplicaCount	**5**	Must be at least minReplicaCount
cooldownPeriod	**300** seconds	Wait before going from **1** to **0** pods; only with a minimum of **0**

EXAM TRAP

With minReplicaCount at **0**, never activate on a metric the pods produce. Once pods reach zero, that metric stops and the deployment never scales up.

Validate before promoting

Shadow deployments test a new serving component before production. They run long-lived shadow variants, so any candidate part of the serving stack is proven first.

Key takeaways

- Inference component copies can scale to zero to make room for others.
- MIG partitioning lets several inference workloads share one GPU.
- DJL Serving's dynamic batching raises throughput.

- Scaling to zero needs a non-CPU/memory scaler and an activation metric the pods don't produce.

Sources: Deploying customized models - Amazon SageMaker AI (opens in a new tab) · Deploying models on Amazon SageMaker HyperPod - Amazon SageMaker AI (opens in a new tab) · Autoscaling policies for your HyperPod inference model deployment - Amazon SageMaker AI (opens in a new tab) · Deploy models for real-time inference - Amazon SageMaker AI (opens in a new tab) · Deploy models with DJL Serving - Amazon SageMaker AI (opens in a new tab) · Testing models with shadow variants - Amazon SageMaker AI (opens in a new tab)

Design and implement enterprise integration architectures.

Start here

THINK FIRST

How do you plug foundation model capabilities into an existing enterprise without opening security holes between tools, teams and accounts?

Why it matters: Questions describe an enterprise integration or access need and ask which gateway, policy or cross-account setting meets it securely.

Put a managed gateway between agents and enterprise tools, and make every access path explicit. Policies, interceptors and resource-based permissions decide who reaches what, including across AWS accounts.

Key terms

Term	Meaning
Gateway target	The link between a gateway and a tool type, such as a Lambda function or REST API service.
Cedar policy	The authorization rule language a gateway policy engine uses.
Interceptor	Gateway logic that checks requests, for example to allow specific MCP operations.
Resource-based policy	A policy attached to a memory that lets principals in another account call it.

The gateway as entry point

An AgentCore gateway gives agents one MCP entry point to tools. It is a managed service: it takes care of authentication and calls your endpoints as tools. A connected agent can call those tools and use a Bedrock model to answer. Each new gateway gets a workload identity automatically.

Targets and templates

Targets connect a gateway to Lambda functions and REST API services. Integration provider templates are added as targets in the console, and you host the server yourself. Pass a path's `{variable}` in the JSON RPC request's `params` field. The Bedrock Runtime target authenticates with the gateway service role.

EXAM TRAP

A template target does not expose every provider API. AgentCore supports only a subset of the APIs the provider offers.

Gateway creation settings

Semantic search can be enabled only when the gateway is created. Without a customer-managed KMS key, AWS encrypts the gateway with an AWS-managed key. A JWT authorizer needs your identity provider's discovery URL and the allowed client IDs.

EXAM TRAP

An explicit gateway service role ARN is not created for you. It must name a role that already exists.

Authorizing tool access

Policy engines apply Cedar policies in log-only or enforcing mode. For IAM-authenticated gateways, match patterns on `principal.id` to limit accounts, roles or users. Interceptors check permissions for specific MCP operations, like `tools/list` and `tools/call`. REQUEST interceptors run before the gateway calls the target.

EXAM TRAP

With the AgentCore CLI, order matters. Run `agentcore add policy-engine` before `agentcore add gateway` so the CLI can resolve the engine ARN.

Cross-account memory access

Attach a resource-based policy to the memory with PutResourcePolicy. The caller passes the full memory ARN as the `memory-id`. Both the resource policy and the caller's identity policy must allow the call, with no explicit deny. Grant specific roles rather than the account root, and audit with CloudTrail.

EXAM TRAP

Deleting the resource-based policy ends cross-account access. Any later cross-account request returns an AccessDeniedException.

Cross-account model import

The bucket owner's account grants the import execution role access. It does so with a bucket policy. If you submit the import job from the console, also edit its default import execution role so `aws:ResourceAccount` names the bucket owner's account ID.

Setup for cross-account custom model import

Step	Done in	Purpose
Bucket policy	Bucket owner's account	Grants the import execution role access

Step	Done in	Purpose
KMS key policy, for a custom key	Bucket owner's account	Lets the import execution role decrypt objects
Edit the console's default role	User's account	Set `aws:ResourceAccount` to the bucket owner's account ID

Automating ML workflows

SageMaker Pipelines orchestrates and automates ML development. You pay only for the Studio environment and the jobs Pipelines runs, such as training and processing.

Key takeaways

- Decide on gateway semantic search at creation; it cannot be turned on later.
- Cross-account memory calls need both policies to allow them and none to deny.
- Memory can send payloads to S3, SNS or Kinesis in another account, set up at creation.
- Use Cedar policies and interceptors to control which callers reach which tools.

Sources: Create an Amazon Bedrock AgentCore gateway - Amazon Bedrock AgentCore (opens in a new tab) · Fine-grained access control for Amazon Bedrock AgentCore Gateway - Amazon Bedrock AgentCore (opens in a new tab) · Cross-account memory access - Amazon Bedrock AgentCore (opens in a new tab) · Create an agent that uses your AgentCore gateway - Amazon Bedrock AgentCore (opens in a new tab) · Built-in templates from integration providers as targets - Amazon Bedrock AgentCore (opens in a new tab) · Create gateway with Policy Engine - Amazon Bedrock AgentCore (opens in a new tab) · Cross-account access to Amazon S3 bucket for custom model import jobs - Amazon Bedrock (opens in a new tab) · Pipelines - Amazon SageMaker AI (opens in a new tab)

Implement FM API integrations.

Start here

THINK FIRST

How should an application call, route and recover foundation model requests so users get fast and reliable answers?

Why it matters: Questions describe an FM integration's latency, routing or reliability need and ask which endpoint, routing strategy or runtime practice fits.

Every model call passes through a hosting choice, a routing decision and a runtime. Pick real-time endpoints for interactive work, route requests toward cached work or across more Regions, and write agent code that survives snapshot restores.

Key terms

Term	Meaning
Inference component	A unit on a SageMaker AI endpoint holding one model and that model's resource needs.
Cross-Region inference profile	A global or geographic profile, such as US or EU, that routes Bedrock requests across more Regions.
Snapshot	Captured startup state that AgentCore Runtime V2 restores to start each agent instance.
Aggregation mode	The gateway mode set by protocolType MCP, which allows only MCP targets.
LMCache	The cache layer for inference; if it fails to initialize, the system recomputes instead of using the cache.

Real-time endpoints

Real-time inference suits interactive, low-latency workloads. Manage single-model endpoints with the AWS CLI, Boto3, Studio or the SageMaker Python SDK. ModelBuilder's deploy method can create a real-time HTTPS endpoint. The endpoint is ready once `EndpointStatus` shows InService.

Many models, one endpoint

An endpoint can host several inference components, each with one model. After deployment, call a component's model directly with InvokeEndpoint. The Maximum instance count setting caps how far the endpoint scales up when traffic grows.

Routing requests in HyperPod

HyperPod intelligent routing defaults to prefixaware. The L1 KV cache uses CPU memory for fast local reuse, while L2 uses Redis or managed tiered storage for sharing across nodes.

HyperPod routing strategies compared

Strategy	Routes by	Note
prefixaware	Same prompt prefix to the same instance	The default when none is set
session	Same user session to the same instance	Keeps a session on one instance
kvaware	Instance whose KV cache hit rate is highest	Needs `invocationEndpoint` set to `/completions`

EXAM TRAP

Watch for fallbacks. Broken KV-aware routing falls back to round-robin, and a failed LMCache start runs in degraded mode that recomputes.

Resilient Bedrock routing

Cross-Region profiles spread Bedrock requests across more Regions. Use global or geographic profiles when processing need not stay in one Region or geography. The exam guide also lists Step Functions for

content-based routing to specialized FMs.

EXAM TRAP

An inference profile does not keep a retired model alive. After its End-of-Life date, the model is no longer available through the profile.

Snapshot startup on Runtime V2

AgentCore Runtime V2 starts agents by restoring a snapshot. Startup work is captured once and shared by every restored instance. Each agent version gets one snapshot until you create or update the agent. Initialization must finish within **120** seconds, or the health check fails.

Snapshot-safe agent code

Restored instances share startup state, so avoid per-instance assumptions. Snapshot-safe cryptographic library builds do not reseed after a restore. In a build warmup hook, the memory and disk calls share **120** seconds, with memory first.

EXAM TRAP

Every restored instance reports hostname localhost and PID **1**, so neither works as an instance ID. Refresh credentials when they expire, since startup credentials can lapse first.

Models in Bedrock Agents

Bedrock Agents works with every Bedrock model, but only some are optimized. For a model without native tool use, define tools in the prompt and write a custom parser. No more models will be optimized, and AWS recommends evaluating AgentCore for new agents.

Gateway protocol and auth modes

Setting `protocolType` to MCP limits a gateway to MCP targets. Leave the field out to allow both MCP and HTTP targets. With AUTHENTICATE_ONLY, the gateway checks the SigV4 signature but makes no authorization decision.

Key takeaways

- A SageMaker AI endpoint serves real-time traffic only after it reaches InService.
- HyperPod routes by prompt prefix unless you pick session or kvaware.
- Runtime V2 agents must be healthy within **120** seconds of startup.
- Never use hostname or PID to identify a restored agent instance.

Sources: Deploy models for real-time inference - Amazon SageMaker AI (opens in a new tab) · KV caching and intelligent routing - Amazon SageMaker AI (opens in a new tab) · Inference profiles with limited model selection - Amazon Bedrock (opens in a new tab) · Optimize your agent for Amazon Bedrock AgentCore Runtime V2 - Amazon Bedrock AgentCore (opens in a new tab) · Deploy models for real-time inference - Amazon SageMaker AI (opens in a new tab) · Create an AgentCore gateway using the API - Amazon Bedrock AgentCore (opens in a new tab) · Single-model endpoints - Amazon SageMaker AI (opens in a new tab) · Exam

Guide (AIP-C01) AWS Certified Generative AI Developer - Professional (opens in a new tab)

Implement application integration patterns and development tools.

Start here

THINK FIRST

What makes a GenAI application easy to build, connect and troubleshoot when calls to models and tools start failing?

Why it matters: Questions describe an app's interface, setup or error and ask which configuration, API or troubleshooting step solves it.

Build on a small set of required settings, then read errors carefully. A **429** and a **503** have different causes. Gateways offer switches for debugging, policy testing and access control.

Key terms

Term	Meaning
ThrottlingException	A Bedrock error with HTTP **429**: the request went over the account quotas.
Idle connection timeout	The fixed **350**-second limit after which NAT gateways, VPC endpoints and NLBs drop idle connections.
Agent resource role	ARN of a service role allowed to call the agent's API operations.
AUTHENTICATE_ONLY	A gateway authorizer type that verifies SigV4 signatures but leaves authorization to others.

Reading Bedrock error codes

A 429 points to quotas; a 503 does not. For ResourceNotFound, call ListFoundationModels to see which models you can use. For an overloaded_error, retry with backoff and jitter, and honor any Retry-After header.

Common Amazon Bedrock errors

Error	HTTP code	Cause
ThrottlingException	**429**	The request exceeded the account quotas
ServiceUnavailable	**503**	Not related to account quotas or rate limits
FTUFormNotFilled	**404**	Model use case details were never submitted

Web Search API errors

InvokeWebSearch rejects a `searchQuery` over 200 characters. It also rejects `maxResults` outside **1** to **25**. On a **429**, retry with exponential backoff or request a higher quota. On a **503** with `retryAfterSeconds`, wait that many seconds first.

EXAM TRAP

Advanced Prompt Optimization templates need double curly brackets. Writing `{variable}` instead of `{{variable}}` returns a ValidationException.

Idle connections that drop

NAT gateways, VPC endpoints and NLBs end idle connections at 350 seconds. To keep long calls alive, enable TCP keep-alive in the SDK. In boto3, `tcp_keepalive` in the Config object defaults to False, so set it to True.

EXAM TRAP

SDK keep-alive alone has no effect while Linux keeps `tcp_keepalive_time` at its **7200**-second default, which outlasts the idle timeout.

Preparing an agent

Preparing an agent takes a resource role, a model and instructions. With no action groups or knowledge bases, it answers from the FM, instructions and base templates. For cross-Region inference, put an inference profile ID in `foundationModel`. By default, history ends after **30** idle minutes.

EXAM TRAP

The service role Bedrock creates for you lacks permissions for features that are still in preview.

Creating a gateway

CreateGateway needs at least `name`, `roleArn` and `authorizerType`. CUSTOM_JWT also needs an `authorizerConfiguration`. AUTHENTICATE_ONLY checks the caller's SigV4 signature and leaves authorization to the target or a policy engine. Setting `protocolType` to MCP allows only MCP targets.

Debugging a gateway

Set `exceptionLevel` to DEBUG to get debugging messages. The AgentCore CLI sets it to NONE by default; pass `--exception-level debug` at creation. Run a policy engine in LOG_ONLY to test policies before switching to ENFORCE.

EXAM TRAP

Interceptors run on every request. An explicit gateway service role must allow `lambda:InvokeFunction` on the interceptor function.

Building blocks for apps

Console-created flows start with a single prompt node. CreateSession returns a session ID and ARN for recording events with CreateInvocation and PutInvocationStep. The exam guide lists Amplify UI components, OpenAPI specifications and Prompt Flows for accessible interfaces.

Orchestrating ML steps

Step Functions runs multi-step ML workflows on SageMaker AI infrastructure. You write the workflow in Python and skip provisioning resources separately. Gateways can also find tools by meaning once `searchType` is `SEMANTIC`.

Key takeaways

- A **503** from Bedrock is not a quota issue; quota limits return a **429**.
- Keep-alive needs both the SDK setting and an OS interval under the **350**-second idle timeout.
- Agents keep conversation history for **30** idle minutes by default.
- Use DEBUG exception level and LOG_ONLY policies while troubleshooting a gateway.

Sources: Create and configure agent manually - Amazon Bedrock (opens in a new tab) · Create an AgentCore gateway using the API - Amazon Bedrock AgentCore (opens in a new tab) · Troubleshooting Amazon Bedrock API Error Codes - Amazon Bedrock (opens in a new tab) · Troubleshoot AgentCore Web Search - Amazon Bedrock AgentCore (opens in a new tab) · Troubleshooting - Amazon Bedrock (opens in a new tab) · Create and design a flow in Amazon Bedrock - Amazon Bedrock (opens in a new tab) · Create a session to prepare to store conversation history and context - Amazon Bedrock (opens in a new tab) · Exam Guide (AIP-C01) AWS Certified Generative AI Developer - Professional (opens in a new tab)

Domain 3 | 20% of the exam | 17 min read

AI Safety, Security, and Governance

Overview

This domain is about 20% of the exam, roughly 15 of its 75 questions.

Every point below comes from AWS's official documentation; each section lists its sources.

Implement input and output safety controls.

Start here

THINK FIRST

How do you stop harmful input from reaching a model, and harmful or malformed output from reaching your users?

Why it matters: Questions describe a safety, format or misuse risk and ask which guardrail action, output control or security setting handles it.

Safety comes in layers placed before, inside and after the model. Guardrails screen prompts and responses, and structured outputs force a predictable shape. Zero operator access and IAM condition keys protect your data and access.

Key terms

Term	Meaning
inputAction and outputAction	Guardrail settings for what happens when harmful content shows up in prompts or responses.
Structured outputs	A Bedrock capability that makes responses conform to your JSON schemas and tool definitions.
Strict tool use	Setting `strict: true` on a tool so its name and inputs are validated against the schema.
Zero operator access	Bedrock's data security model, subject to model-specific abuse and CSAM detection.
Confused deputy	A cross-service access risk; condition keys limit which resource or account may be tied to that access.

Layers of defense

Defense in depth stacks controls before, inside and after the model. The exam guide pairs Comprehend pre-processing filters and Bedrock guardrails with Lambda post-processing and API Gateway response filtering. It adds jailbreak detection, input sanitization and adversarial testing. Text-to-SQL transformations help keep outputs deterministic.

Guardrail actions

Set separate runtime actions for prompts and responses. Use `inputAction` for prompts and `outputAction` for responses. Available actions include Block and Detect, and Mask works with sensitive information filters only.

Amazon Bedrock guardrail actions compared

Action	Effect on content	What you see
Block	Blocked	Blocked messaging in its place
Mask	Anonymized; sensitive information filters only	Identifier tags such as {NAME} or {EMAIL}
Detect	No action taken	Detections in the trace response

EXAM TRAP

Filter strength is not a confidence threshold you must exceed. At strength HIGH, a guardrail blocks content even when its evaluation returns LOW confidence.

Structured outputs

Structured outputs make responses follow your JSON schema. Responses always parse as JSON, fields keep their declared types and required fields never go missing. Claude offers two mechanisms, JSON Schema output format and strict tool use, usable alone or together.

Schema rules

Every schema object must set `additionalProperties` to false. A schema allows at most **12** optional properties across all nested objects. Compiled grammars stay cached for **24** hours from first access. The Converse API takes the schema in `outputConfig.textFormat`.

EXAM TRAP

With Claude through InvokeModel, `output_config.format` takes the schema as a JSON object, not a JSON string. Enabling citations alongside structured outputs returns a **400** error.

Strict tool use

Add `strict: true` to validate tool names and inputs. The tool input must then match the schema exactly, and the model cannot add fields outside it. Structured outputs span the Sonnet, Opus, Haiku, Fable

and Mythos families, though support varies by model and endpoint.

Abuse and CSAM detection

Bedrock may run automated abuse detection to catch policy violations. If it finds apparent CSAM in image inputs or outputs, it may block the request with a **400** ValidationException. Model providers cannot reach Bedrock's Model Deployment Accounts, so they never see your logs, prompts or completions.

Securing access

Stop confused deputies with `aws:SourceArn` and the full resource ARN. Choose `aws:SourceArn` to allow one resource, or `aws:SourceAccount` to allow any resource in that account. Bedrock requires TLS **1.2** and recommends TLS **1.3**.

EXAM TRAP

Keep confidential data out of tags and free-form name fields. That text may end up in billing or diagnostic logs.

Key takeaways

- Mask works only with sensitive information filters; Detect only reports in the trace.
- A HIGH filter strength blocks content even at LOW confidence.
- Structured outputs need `additionalProperties: false` on every object and break with Anthropic citations.
- Model providers cannot access Bedrock logs or your prompts and completions.

Sources: Exam Guide (AIP-C01) AWS Certified Generative AI Developer - Professional (opens in a new tab) · Options for handling harmful content detected by Amazon Bedrock Guardrails - Amazon Bedrock (opens in a new tab) · Get validated JSON results from models - Amazon Bedrock (opens in a new tab) · Amazon Bedrock abuse detection - Amazon Bedrock (opens in a new tab) · Cross-service confused deputy prevention - Amazon SageMaker AI (opens in a new tab) · Structured outputs for Anthropic Claude models - Amazon Bedrock (opens in a new tab) · Data protection - Amazon Bedrock (opens in a new tab)

Implement data security and privacy controls.

Start here

THINK FIRST

How do you keep sensitive data out of prompts, responses, logs and storage when you build on Amazon Bedrock?

Why it matters: Questions describe a workload that handles sensitive data and ask which Bedrock control or setting protects it.

Protecting a foundation model workload takes layers: identity and network controls around Bedrock, then filters on the data itself. Guardrails can block or mask PII in text. Masking does not reach every copy of the data, such as invocation logs or tool fields.

Key terms

Term	Meaning
Sensitive information filter	A guardrail filter that uses context-aware machine learning to find PII in prompts and responses.
Mask mode	The guardrail swaps detected PII for its type, such as {NAME} or {EMAIL}.
Model Deployment Account	An AWS account per model provider in each Region, which the provider cannot access.
Data access policy	An OpenSearch Serverless policy that limits a collection to the knowledge base service role.

Shared responsibility

AWS secures the cloud; you secure what you put in it. AWS protects the infrastructure that runs its services. You judge how sensitive your data is and which company rules and laws apply.

Identity, transport and keys

Every Bedrock request is signed and sent over TLS. Sign requests with an access key tied to an IAM principal, or with temporary AWS STS credentials. Bedrock requires **TLS 1.2** and recommends **TLS 1.3**. Start custom policies with minimum permissions and add more only as needed. A guardrail uses an AWS managed key by default, or your own customer-managed KMS key.

Model provider isolation

Model providers cannot see your prompts or completions. Each Region has **one** Model Deployment Account per model provider. Providers have no access to these accounts. So they cannot read Bedrock logs or customer prompts and completions.

> **EXAM TRAP**
> Using a third-party model does not hand your prompts to that model's vendor. Providers have no access to the deployment accounts that serve the model.

Private knowledge base storage

A private OpenSearch Serverless collection is reached only through PrivateLink. Edit the collection's network access policy so Amazon Bedrock is an allowed service. Choose AWS service private access and enter `bedrock.amazonaws.com`. A data access policy then limits the collection to the knowledge base service role.

> **EXAM TRAP**
> A custom service role is not set up for you. Only when Bedrock creates the role can you skip the extra security configuration for the new knowledge base.

Block versus mask

A guardrail can block the whole message or mask the PII. The filter is probabilistic and reads context, so it is not an exact pattern match. You can add a custom regex, but it cannot use a lookaround match.

What each sensitive information action does

Feature	Block	Mask
When PII is found	Blocks all the content	Replaces the PII with its type
What the user gets	A message you configure	Text with placeholders such as {EMAIL}

EXAM TRAP

Do not expect the filter to inspect tool use. It evaluates text only, so PII in tool arguments, tool results or tool definitions is neither blocked nor masked.

Where masking stops

Masking changes the reply, not every copy of the data. With invocation logging on, the input field in CloudWatch Logs keeps the original request. Protect those logs with CloudWatch log data protection. The `match` field in the guardrail trace also holds the original PII value.

EXAM TRAP

Turning on Mask mode does not clean your invocation logs. Their input field always keeps the unmodified request, whatever the guardrail did.

Redaction in Data Automation

Bedrock Data Automation redacts PII only after you turn it on. PII detection is off by default. Enable it with modality-specific overrides in `overrideConfiguration`. Redaction changes only the JSON results, never your input assets or blueprints. The `ALL` entity type covers every supported PII type, and `AWS_SECRET_KEY` is one of them.

Names, tags and privacy tools

Keep PII out of tags, names and other free-form fields. Text in tags or name fields may end up in billing or diagnostic logs. For agents, avoid PII in fields without customer managed key support, such as action group names. Amazon Comprehend and Amazon Macie detect PII, and S3 Lifecycle rules enforce data retention.

Key takeaways

- Mask mode swaps PII for its type, like {NAME}; Block mode rejects the whole message with your own reply.
- Guardrail masking never changes invocation logs; protect them with CloudWatch log data protection.
- The sensitive information filter skips tool arguments, tool results and tool definitions.
- Data Automation PII redaction is off by default and changes only the JSON results.

- A private OpenSearch Serverless collection needs a PrivateLink endpoint and a network policy that allows Bedrock.

Sources: Infrastructure security in Amazon Bedrock - Amazon Bedrock (opens in a new tab) · Set up security configurations for your knowledge base - Amazon Bedrock (opens in a new tab) · Remove PII from conversations by using sensitive information filters - Amazon Bedrock (opens in a new tab) · Data protection - Amazon Bedrock (opens in a new tab) · Security, Guardrails, and Observability in Amazon Bedrock - Amazon Bedrock (opens in a new tab) · Preventative security best practice for agents - Amazon Bedrock (opens in a new tab) · Sensitive data detection and redaction in Amazon Bedrock Data Automation - Amazon Bedrock (opens in a new tab) · Types of PII that Amazon Bedrock Data Automation can detect and redact - Amazon Bedrock (opens in a new tab)

Implement AI governance and compliance mechanisms.

Start here

THINK FIRST

When an auditor asks how a model was built and who oversees it, which records and controls let you answer?

Why it matters: Questions describe an audit, oversight or traceability need and ask which AWS service or setting meets it.

Governance means systematic visibility into how models are built, validated and used. Lineage tracking records each workflow step, model cards record intent and risk, and logs and metrics record what happens at run time.

Key terms

Term	Meaning
Model governance	A framework for systematic visibility into ML model development, validation and usage.
Model card	An immutable record of a model's intended uses, risk ratings, training details and evaluation results.
Lineage graph	A map of an ML workflow that SageMaker AI builds automatically, from training to deployment.
Trial component	The lineage entity that tracks a training job and holds all of its parameters.
Unified telemetry	An AgentCore mode that keeps an agent's spans and logs together in one log group.

Tracing a model's lineage

Lineage Tracking records each workflow step from data to deployment. It stores training jobs, datasets, pipelines, endpoints and the models themselves. A trial component holds every parameter of a training job. Query the lineage at any time to find the exact artifacts that trained a model. These records support auditing

and compliance checks.

EXAM TRAP

You do not draw the lineage by hand. SageMaker AI creates the lineage graph automatically for an end-to-end training and deployment workflow.

Documenting intent and risk

Model Cards keep one immutable record of a model's purpose and risk. Risk managers, data scientists and ML engineers record intended uses, risk ratings, training details and evaluation results. The record covers the model from conception to deployment. Programmatic model cards are a building block of compliance frameworks for FM deployments.

One view of every model

Model Dashboard brings model information into one unified view. It pulls in Model Monitor, Transform Jobs, Endpoints, ML Lineage Tracking and CloudWatch data. Use it to see high-level model information and track performance.

Three SageMaker governance features compared

Feature	Model Cards	Lineage Tracking	Model Dashboard
Main job	Document and share essential model information	Record the steps of an ML workflow	Track model performance in one view
What it holds	Intended uses, risk ratings, training details, evaluation results	Training jobs, datasets, pipelines, endpoints, models	Data from Model Monitor, Endpoints, CloudWatch and more

Least-privilege personas

Role Manager builds persona-based IAM roles with least privilege. Administrators define least-privilege permissions for common machine learning activities. Each role matches a persona's business needs. To share assets in SageMaker Studio, administrators set up databases and ML infrastructure with Amazon DataZone.

Tracking data sources

Catalog where data comes from and tag what the model writes. Register each dataset's origin in the AWS Glue Data Catalog. Metadata tagging attributes FM-generated content to that origin. CloudTrail adds audit logging, AWS Glue can track data lineage automatically, and CloudWatch Logs collects decision logs.

EXAM TRAP

CloudTrail is the audit log, not the data registry. Registering where data comes from is the job of the AWS Glue Data Catalog.

Watching Bedrock at run time

CloudWatch metrics, CloudTrail and invocation logging observe Bedrock traffic. These options cover traffic to the `bedrock-runtime` endpoint. To hear about state changes, configure an SNS topic, subscribe to it and create an EventBridge rule.

EXAM TRAP

An SNS topic alone sends nothing on a state change. You also need a subscription and an EventBridge rule.

Agent telemetry modes

Unified telemetry keeps an agent's spans and logs in one log group. One log group lets you write IAM policies and customer managed key encryption for a single agent. Switch modes with `UNIFIED_TRACES_DESTINATION_ENABLED`. ADOT, the AWS OpenTelemetry distro, exports the agent's spans. Unified mode needs ADOT **0.18.0** or later, and evaluations need Transaction Search in both modes.

Unified and split telemetry for AgentCore agents

Feature	Unified	Split
Variable value	`true`	`false`
Where spans go	One log group with the agent's logs	The shared `aws/spans` log group
Default for	Agents created on or after **July 20, 2026**	Agents created before that date

EXAM TRAP

Switching modes does not move old telemetry. Spans already delivered remain in their original log group.

Fair compute across teams

HyperPod task governance shares cluster compute through Kueue. Quota management, priority scheduling and resource sharing policies keep allocation fair across teams. A Compute allocation needs Kubernetes RBAC in a namespace named `hyperpod-ns-team-name`. Interactive Spaces are the development workloads that share the cluster. A high-weight priority class, such as **100**, lets them preempt lower-priority jobs.

EXAM TRAP

Flexible Resource Sharing still lends idle quota; it only disables borrowing. Strict Resource Control is the strategy that disables both lending and borrowing.

Key takeaways

- Lineage Tracking finds the exact artifacts that trained a model, for audits and compliance.
- Model Cards are an immutable record of intended uses, risk ratings and evaluation results.

- The Glue Data Catalog registers where data comes from; CloudTrail provides audit logging.
- AgentCore agents created on or after **July 20, 2026** use unified telemetry by default.
- Changing the telemetry mode never moves spans that were already delivered.

Sources: Model governance to manage permissions and track model performance - Amazon SageMaker AI (opens in a new tab) · Amazon SageMaker ML Lineage Tracking - Amazon SageMaker AI (opens in a new tab) · Monitor the bedrock-runtime endpoint - Amazon Bedrock (opens in a new tab) · Telemetry setup and delivery - Amazon Bedrock AgentCore (opens in a new tab) · Exam Guide (AIP-C01) AWS Certified Generative AI Developer - Professional (opens in a new tab) · Example Create a rule to handle Amazon Bedrock state change events - Amazon Bedrock (opens in a new tab) · Task governance for Interactive Spaces on HyperPod - Amazon SageMaker AI (opens in a new tab) · Create policies - Amazon SageMaker AI (opens in a new tab)

Implement responsible AI principles.

Start here

THINK FIRST

How can you show that model answers follow your rules, judge them for fairness, and enforce policy when agents act?

Why it matters: Questions describe a transparency, fairness or policy need and ask which Bedrock or AgentCore feature meets it.

Responsible AI here rests on three habits: explain outputs, measure fairness and enforce policy. Automated Reasoning checks test answers against formal rules, evaluations score outputs with people or an LLM judge, and Policy in AgentCore decides every tool call.

Key terms

Term	Meaning
Automated Reasoning finding	A result holding the translated premises and claims, a confidence score, the variable values and the rules behind it.
TRANSLATION_AMBIGUOUS	The finding when the translating LLMs disagree on meaning; it lists the alternative interpretations.
Work team	Workers you choose for human evaluation, such as employees or industry subject-matter experts.
LLM-as-a-Judge	Scoring outputs with a model; AgentCore Evaluations uses it for built-in and custom evaluators.
Forbid-wins	Policy engine rule where a matching forbid beats any permit, on top of default-deny.

Transparent outputs

Show users the reasoning and evidence behind an answer. Bedrock agent tracing provides reasoning traces. CloudWatch can collect confidence metrics that quantify uncertainty. Model cards document a foundation model's limitations, and Lambda functions can run automated compliance checks.

Reading Automated Reasoning results

Each finding holds the translated claims, a confidence score and rules. Findings below the confidence threshold become `TRANSLATION_AMBIGUOUS`. When you combine findings, that is the worst result, ahead of IMPOSSIBLE, INVALID, SATISFIABLE and VALID.

Three Automated Reasoning findings compared

Feature	VALID	INVALID	TRANSLATION_AMBIGUOUS
Meaning	Claims proven correct under the rules	Claims contradict the rules	Translating LLMs disagreed
Extra detail	`supportingRules` and `claimsTrueScenario`	`contradictingRules`	The alternative interpretations

EXAM TRAP

TRANSLATION_AMBIGUOUS is not a rule violation. Vague or incomplete variable descriptions are its number one cause, so improve those first.

Testing a policy

Test rules with scenarios first, then add QnA tests. Scenarios check rule correctness; QnA tests check translation accuracy. Through the API, start a `GENERATE_POLICY_SCENARIOS` build workflow. Then call `GetAutomatedReasoningPolicyNextScenario` once per scenario. Split a genuinely complex domain into several focused policies.

EXAM TRAP

A rule without an if-then structure is not a harmless note. It becomes an axiom, a statement that is always true.

Human evaluation teams

A Bedrock work team holds at most 50 workers. Workers can be your own employees or subject-matter experts from your industry. Workers hear about an assigned job only at the moment you first add them to a work team.

EXAM TRAP

A new evaluation job does not notify workers already on the team. The notice goes out only when a worker is first added.

Fairness and agent evaluation

Compare prompts with A/B tests and score outputs with an LLM judge. Prompt Management and Prompt Flows support systematic A/B testing for fairness. Online evaluation can sample a share of sessions, such as **10%**, or use filters. Batch jobs accept ground truth such as expected tool trajectories.

On-demand and batch agent evaluation compared

Feature	On-demand	Batch
What you supply	The span or trace IDs to score	A CloudWatch Logs location and the evaluators
Who collects spans	You, then you call Evaluate per session	The service, in one asynchronous job

Enforcing policy on tools

Policy in AgentCore checks each tool call at the AgentCore Gateway. The engine evaluates every applicable policy for each request. It enforces default-deny and forbid-wins automatically. Each decision is logged in CloudWatch metrics and logs, so compliance teams can audit it.

EXAM TRAP

A permit does not override a matching forbid. The engine applies forbid-wins semantics on every tool invocation.

Checking policies before use

Automated reasoning flags policies that always allow or always deny. Cedar is a policy language, and Dogwood is a superset of it: every valid Cedar policy is also valid Dogwood. Cedar analysis spots permits with no limiting conditions and forbids with no exceptions. Natural language authoring validates candidate policies against the tool schema. A guardrail's confidence score can permit, forbid or suppress a response.

Key takeaways

- INVALID findings list `contradictingRules`; VALID findings list `supportingRules`.
- Low-confidence findings surface as TRANSLATION_AMBIGUOUS, the worst severity when you combine results.
- A Bedrock human evaluation work team holds at most **50** workers.
- Policy in AgentCore is default-deny, a forbid always wins, and every decision is logged.

Sources: Test an Automated Reasoning policy - Amazon Bedrock (opens in a new tab) · Exam Guide (AIP-C01) AWS Certified Generative AI Developer - Professional (opens in a new tab) · Evaluate agent performance with Amazon Bedrock AgentCore Evaluations - Amazon Bedrock AgentCore (opens in a new tab) · Manage a work team for human evaluations of models in Amazon Bedrock - Amazon Bedrock (opens in a new tab) · Policy in Amazon Bedrock AgentCore: Control Agent Interactions - Amazon Bedrock AgentCore (opens in a new tab) · Core concepts - Amazon Bedrock AgentCore (opens in a new tab) · Automated Reasoning policy best practices - Amazon Bedrock (opens in a new tab) · Evaluation types - Amazon Bedrock AgentCore (opens in a new tab)

Domain 4 | 12% of the exam | 11 min read

Operational Efficiency and Optimization for GenAI Applications

Overview

This domain is about 12% of the exam, roughly 9 of its 75 questions.

Every point below comes from AWS's official documentation; each section lists its sources.

Implement cost optimization and resource efficiency strategies.

Start here

THINK FIRST

How do you cut foundation model costs without giving up answer quality or response speed?

Why it matters: Questions describe a workload's traffic and budget and ask which pricing, hosting or caching choice lowers its cost.

Cost falls when you pay less per token, send fewer tokens and keep hardware busy. Pick the pricing mode that fits the traffic, trim tokens per request, and share or cache work so fewer model calls run.

Key terms

Term	Meaning
Batch inference	Predictions on a batch of observations, for large datasets or when no immediate response is needed.
SageMaker AI Savings Plan	A plan for consistent SageMaker AI usage that can cut costs by up to **64%**.
Multi-model endpoint	One endpoint whose instance serves many models and caches the frequently used ones.
KV caching	HyperPod Inference caching with an L1 tier in CPU memory and an L2 tier in Redis or tiered storage.
Effort parameter	A Claude Opus 4.5 setting for how liberally tokens are spent; it defaults to `high`.

Bedrock pricing modes

Batch suits work that can wait; the Reserved Tier suits steady high load. Cross-Region inference with global profiles also lowers token costs for some models.

Two discounted Bedrock options compared

Feature	Batch inference	Reserved Tier
Saving	**50%** versus on-demand	**30-50%** with a commitment
Best fit	Large datasets with no need for an immediate answer	Steady high-volume load
Catch	Jobs time out after **24** hours by default, up to **168**	Requires a commitment

EXAM TRAP

Long Claude Sonnet 4 requests are not billed in two parts. Above **200K** input tokens, the long context rate covers the entire request.

Spending fewer tokens

Track tokens per request and trim prompts first. The `token-efficient-tools-2025-02-19` beta header for tool use works with Claude 3.7 Sonnet and Claude 4+. For Claude Opus 4.5, the effort parameter defaults to `high`. Setting `medium` or `low` makes token use more conservative.

Choosing a cost-effective model

Send each query to a model tier that matches its complexity. Measure the price-to-performance ratio and weigh inference cost against response quality. Inference Recommender compares instances so you can see model performance and cost together.

Sharing instances

Multi-model and multi-container endpoints share one instance. Frequently used models stay cached in memory and on disk until space runs out. GPU-backed multi-model endpoints need the NVIDIA Triton Inference Server. Run them on more than **one** instance for availability and cache hits.

EXAM TRAP

Model caching does not always pay off. When many models are each called once or rarely, setting `ModelCacheSetting` to `Disabled` allows higher TPS.

Optimizing the model itself

Inference optimization techniques depend on the hardware. Most GPU instances support quantization and speculative decoding. Speculative decoding jobs add no cost, while quantization and compilation jobs bill for the instance. Concurrency runs several model copies on one device, and a stopping condition caps job cost.

EXAM TRAP

Compilation is not the GPU option here. It is the technique for custom silicon such as AWS Inferentia instances.

Key takeaways

- Bedrock batch inference saves **50%** versus on-demand when you can wait for results.
- The Reserved Tier suits steady high-volume load, saving **30-50%** with a commitment.
- Speculative decoding jobs cost nothing extra; quantization and compilation jobs bill for the instance.
- HyperPod intelligent routing defaults to `prefixaware` and supports only vLLM-based containers.

Sources: Capacity and Performance - Amazon Bedrock (opens in a new tab) · Inference cost optimization best practices - Amazon SageMaker AI (opens in a new tab) · Exam Guide (AIP-C01) AWS Certified Generative AI Developer - Professional (opens in a new tab) · Create an inference optimization job - Amazon SageMaker AI (opens in a new tab) · Set SageMaker AI multi-model endpoint model caching behavior - Amazon SageMaker AI (opens in a new tab) · KV caching and intelligent routing - Amazon SageMaker AI (opens in a new tab) · Request and Response - Amazon Bedrock (opens in a new tab) · Create a Multi-Model Endpoint - Amazon SageMaker AI (opens in a new tab)

Optimize application performance.

Start here

THINK FIRST

When a generative AI app is slow or throttled, which levers improve latency and throughput, and what does each one cost?

Why it matters: Questions describe a latency, throughput or retrieval problem and ask which Bedrock or SageMaker setting fixes it.

Performance work balances latency, throughput and cost. Latency-optimized models suit time-sensitive apps, service tiers trade cost against throttling risk, and reranking makes retrieved documents more relevant.

Key terms

Term	Meaning
Latency-optimized inference	A preview Bedrock mode you request by setting the latency parameter to `optimized`.
Model Unit (MU)	A unit defined by the input tokens it can process across all requests in **one** minute.
Service tier	An on-demand class, Flex, Standard or Priority, with its own cost, limits and throttling risk.

Term	Meaning
Reranking model	A model that reorders retrieved documents by relevance during Retrieve or RetrieveAndGenerate.

Latency-optimized inference

Set the latency parameter to `optimized`; otherwise requests run standard. This feature is in preview and may change. For Llama 3.1 405B it supports up to **11K** total tokens per request. Larger requests fall back to standard mode. Its CloudWatch metrics appear under `model-id+latency-optimized`.

EXAM TRAP

Reaching the latency optimization quota does not fail the request. Bedrock tries to serve it at standard latency and charges standard rates.

On-demand service tiers

Start on Standard; move to Priority only when throttling hurts users. Watch CloudWatch throttle metrics to decide. Dev and test environments can drop down to Flex.

Flex, Standard and Priority on-demand tiers

Feature	Flex	Standard	Priority
Requests per minute	**10-100**	**100-500**	**500-1000+**
Tokens per minute	**5K-50K**	**50K-150K**	**150K-300K+**
Throttling risk	High	Medium	Low
Trade-off	Lowest cost per token, no SLA	The starting point for most workloads	Highest on-demand cost, enhanced SLA

Reserved capacity

Move to the Reserved tier once steady load tops 40% of on-demand costs. Begin with a **1-month** commitment. The minimum purchase is **1** model unit. Within the capacity you buy, there is no RPM throttling.

Provisioned Throughput

Provisioned Throughput is billed hourly for each one you purchase. With a **6-month** commitment, you cannot delete it until the six-month commitment term is over.

EXAM TRAP

A customized model does not run on on-demand capacity. You must purchase Provisioned Throughput before you can use it.

Batch and offline work

Batch jobs wait behind on-demand traffic. On-demand requests take priority, so batch processing time depends on spare capacity. Completion is not guaranteed. A batch job that uses a VPC must be submitted through the API, not the console.

Faster, more relevant retrieval

Rerank, decompose and tune retrieval to feed better context. A reranking model reorders documents returned by Retrieve or RetrieveAndGenerate. `AgenticRetrieveStream` splits complex queries into sub-queries and retrieves iteratively. The Retrieve operation lets you decouple and customize RAG steps. Index optimization, query preprocessing and hybrid search with custom scoring also help.

Agent orchestration latency

An agent with one knowledge base can make fewer LLM calls. When the single knowledge base conditions are met, Bedrock reduces the LLM calls needed per request.

> **EXAM TRAP**
>
> Overriding the default orchestration prompt template has a cost. The agent switches to the standard flow, which may add LLM calls and latency.

Tuning and right-sizing

Tune parameters per use case and right-size the hosting. Adjust temperature and top-k or top-p, and A/B test each change. Check CloudWatch CPU and GPU use to find idle endpoints, then combine their models. SageMaker Neo can optimize models for fewer or smaller instances. Asynchronous inference runs a fixed instance count and can scale to zero.

Key takeaways

- Latency-optimized inference is a preview you opt into per request; standard is the default.
- Priority has the lowest throttling risk and highest cost; Flex has the lowest cost and no SLA.
- Custom models need Provisioned Throughput, which is billed hourly.
- Pre-computation, parallel requests and response streaming ease latency-cost trade-offs.
- Profile prompt-completion patterns and optimize vector queries to speed up whole workflows.

Sources: Exam Guide (AIP-C01) AWS Certified Generative AI Developer - Professional (opens in a new tab) · Capacity and Performance - Amazon Bedrock (opens in a new tab) · Retrieving information from data sources using Amazon Bedrock Knowledge Bases - Amazon Bedrock (opens in a new tab) · Optimize model inference for latency - Amazon Bedrock (opens in a new tab) · Increase model invocation capacity with Provisioned Throughput in Amazon Bedrock - Amazon Bedrock (opens in a new tab) · Create a batch inference job - Amazon Bedrock (opens in a new tab) · Optimize performance for Amazon Bedrock agents using a single knowledge base - Amazon Bedrock (opens in a new tab) · Inference cost optimization best practices - Amazon SageMaker AI (opens in a new tab)

Implement monitoring systems for GenAI applications.

Start here

THINK FIRST

How would you find out why an FM app or agent is slow, failing or drifting, from tool calls down to GPU use?

Why it matters: Questions describe a monitoring or troubleshooting need and ask which metric, log or setting reveals the problem.

GenAI monitoring tracks FM-specific signals, such as token usage and hallucination rates, alongside inference framework metrics. AgentCore observability covers runtime agents plus memory, gateway and built-in tools. SageMaker AI detailed observability covers endpoints, and Model Invocation Logs support detailed request and response analysis.

Key terms

Term	Meaning
Transaction Search	A CloudWatch feature you enable once before AgentCore metrics, spans and traces appear.
aws/spans log group	The CloudWatch Logs group that holds full span data for AgentCore built-in tools.
Detailed observability	A SageMaker AI endpoint feature that publishes OTel metrics, including GPU and framework metrics, to CloudWatch.
Golden dataset	A dataset used in FM-specific troubleshooting to detect hallucinations in model output.
TTFT	Time to first token, one of the native vLLM and SGLang metrics.

Turning on agent observability

Enable CloudWatch Transaction Search once to see AgentCore data. Agents in the AgentCore runtime get session metrics automatically. Memory, gateway and built-in tools can be monitored even when agents run elsewhere. Metrics are on by default; signals marked with an asterisk need explicit enablement. Logs can go to CloudWatch Logs, Amazon S3 or Firehose.

EXAM TRAP

Policy data has no tab of its own. It appears under the AgentCore Gateway tab in CloudWatch gen AI observability.

Built-in tool metrics

Tool metrics show latency, throttling and usage for each tool. Latency runs from request receipt until the first response token. Throttles counts requests over transactions per second (TPS) or quota limits, which return `ThrottlingException` with HTTP **429**. Code interpreter and browser metrics are batched each minute. Usage data may lag by up to **60** minutes.

Browser tool troubleshooting

Use the Browser tool for specific page actions, not general search. To avoid CAPTCHA blocks, run general web search through a non-browser MCP tool such as Tavily. Browser sessions time out after **900** seconds by default, set by `sessionTimeoutSeconds`.

> **EXAM TRAP**
> Session Replay and Web Bot Auth fail in a context made with `browser.new_context()`. Their extensions work only in the default browser context.

Endpoint detailed observability

Detailed observability sends OpenTelemetry (OTel) endpoint metrics. On SageMaker AI endpoints, it exposes native vLLM and SGLang metrics, such as TTFT, KV cache utilization and queue depth, with no custom instrumentation. GPU metrics show which inference component uses each GPU. It defaults to on for endpoint configurations created after **June 17, 2026**.

> **EXAM TRAP**
> These are not Prometheus metrics. They are OTel metrics sent to CloudWatch over OTLP, and PromQL is only the query language.

Configuring endpoint metrics

Set the scrape rate and enable OTel enrichment to query metrics. `MetricPublishFrequencyInSeconds` defaults to **60** and accepts **10**, **30**, **60**, **120**, **180**, **240** or **300**. OTel enrichment makes metrics queryable with PromQL in CloudWatch Query Studio and Amazon Managed Grafana. A custom container exposes Prometheus-format metrics on port **8080**, at `/metrics` by default.

> **EXAM TRAP**
> Detailed observability does not replace enhanced metrics. `EnableDetailedObservability` and `EnableEnhancedMetrics` are separate features that can run on the same endpoint.

Dashboards and events

SageMaker AI Insights groups endpoint metrics into three views. The views are Performance, Capacity and Reliability. For Bedrock, an EventBridge rule on `aws.bedrock` with detail-type Batch Inference Job State Change catches a finished job. Route it to a subscribed SNS topic to get notified.

GenAI-specific signals

Track tokens, hallucinations and drift, not only errors. CloudWatch can track token usage, prompt effectiveness, hallucination rates and response quality. Model Invocation Logs support detailed request and response analysis. Golden datasets detect hallucinations, output diffing checks consistency, and reasoning path tracing finds logic errors. For vector stores, monitor performance and validate data quality.

Human evaluation datasets

Custom prompt datasets for human evaluation use JSON Lines. Save the file with the `.jsonl` extension. A single file may hold no more than **1,000** JSON objects.

Key takeaways

- Enable CloudWatch Transaction Search once before AgentCore metrics, spans and traces appear.
- Built-in tool throttling returns `ThrottlingException` with HTTP **429**.
- Detailed observability sends OTel metrics over OTLP; PromQL is only the query language.
- AgentCore Browser sessions time out after **900** seconds by default.
- Golden datasets catch hallucinations, and output diffing checks response consistency.

Sources: Exam Guide (AIP-C01) AWS Certified Generative AI Developer - Professional (opens in a new tab) · Amazon Bedrock AgentCore generated observability data - Amazon Bedrock AgentCore (opens in a new tab) · Amazon SageMaker AI detailed observability for inference endpoints - Amazon SageMaker AI (opens in a new tab) · AgentCore generated built-in tools observability data - Amazon Bedrock AgentCore (opens in a new tab) · Troubleshoot AgentCore Browser - Amazon Bedrock AgentCore (opens in a new tab) · Getting started with detailed observability - Amazon SageMaker AI (opens in a new tab) · SageMaker AI Insights dashboard - Amazon SageMaker AI (opens in a new tab) · Example Create a rule to handle Amazon Bedrock state change events - Amazon Bedrock (opens in a new tab)

Domain 5 | 11% of the exam | 9 min read

Testing, Validation, and Troubleshooting

Overview

This domain is about 11% of the exam, roughly 8 of its 75 questions.

Every point below comes from AWS's official documentation; each section lists its sources.

Implement evaluation systems for GenAI.

Start here

THINK FIRST

How do you show that a model, a RAG pipeline or an agent is good enough, and that it stays good after each change?

Why it matters: Questions describe what needs judging, a model, retrieval or an agent, and ask which evaluation type, metric or setting fits.

GenAI evaluation judges faithfulness, context relevance, coherence and refusals, often with a model as the judge. Bedrock runs evaluation jobs for models and for RAG pipelines, which retrieve texts and then generate a response from them. AgentCore scores live agent sessions and A/B tests.

Key terms

Term	Meaning
Evaluation task type	The task a Bedrock evaluation job asks the model to do; each job has exactly **one**.
Ground truth	A reference response in the prompt dataset; some metrics need it, and human raters never score it.
Faithfulness	A metric for how well responses avoid hallucination compared with the retrieved texts.
Online evaluation	An AgentCore configuration that keeps scoring an agent from live traffic.
Session stickiness	Gateway behavior that keeps in-flight sessions on their assigned target during an A/B test.

Automatic model evaluation

Each automatic job needs a task type, which brings metrics and datasets. You choose only **one** task type per job. Evaluate Bedrock models, or other models by putting their responses in the prompt dataset.

Accuracy in two automatic task types

Feature	Question and answer	Text summarization
Built-in datasets	BoolQ, NaturalQuestions, TriviaQA	Gigaword
Accuracy metric	NLP-F1	BERTScore

EXAM TRAP

When you bring your own responses, their name is not free text. It must match the `modelIdentifier` parameter in the prompt dataset.

Model as a judge

An evaluator model scores responses on built-in or custom metrics. Pick at least **one** built-in metric for the evaluator. When the model under test is a Bedrock model, both models must be in the same Region. These jobs need no CORS configuration. Logical coherence looks for gaps and contradictions; Refusal spots declined requests.

EXAM TRAP

Correctness does not ignore your reference answer. When you supply ground truth, the evaluator model considers it while scoring.

Evaluating RAG

A RAG job is either retrieve only or retrieve and generate. Context coverage needs ground truth in the prompt dataset. Retrieve and generate jobs can also score citations with `Builtin.CitationCoverage` and `Builtin.CitationPrecision`. The IAM policy must allow the evaluator model and any Bedrock response generator.

Three RAG evaluation metrics compared

Feature	Context relevance	Context coverage	Faithfulness
Metric name	`Builtin.ContextRelevance`	`Builtin.ContextCoverage`	`Builtin.Faithfulness`
What it measures	How relevant retrieved texts are to the question	How much retrieved texts cover the ground truth	How well responses avoid hallucination

EXAM TRAP

For your own RAG responses, the name you enter must match `knowledgeBaseIdentifier`, not `modelIdentifier`.

Human evaluation

Human jobs compare up to two response providers. The two can be different types, such as a Bedrock model and your own responses. A custom prompt dataset holds at most **1,000** prompts. The task type does not steer responses, so every instruction goes in the prompts. Ground truth appears as a reference and is never rated.

EXAM TRAP

Outside the console, a human job needs a SageMaker AI flow definition ARN. Its `AwsManagedHumanLoopRequestSource` must be `AWS/Bedrock/Evaluation`.

Online evaluation of agents

AgentCore online evaluation scores live sessions continuously. Use up to **25** evaluators. Sampling defaults to **10%** of sessions. Setting `enableOnCreate` to false starts the configuration as DISABLED. It reads `aws/spans` automatically, but you must list the event log group.

EXAM TRAP

Creating a configuration as ENABLED has a side effect. The custom evaluators you selected are locked automatically.

Agent metrics

Bedrock Agents publish metrics in the `AWS/Bedrock/Agents` namespace. TTFT, the time-to-first-token metric, is emitted only when streaming is enabled for `InvokeAgent` or `InvokeInlineAgent`. Judge agents on task completion rates, tool usage effectiveness and reasoning quality across multi-step workflows.

Quality gates and user feedback

Keep quality steady with continuous evaluation and regression tests. Add automated quality gates to deployments. Before an FM update, run synthetic user workflows and check hallucination rates and semantic drift. Feedback interfaces, rating systems and annotation workflows bring in the user's view.

Key takeaways

- Each Bedrock model evaluation job takes exactly **one** task type.
- Context coverage needs ground truth; faithfulness checks hallucination against retrieved texts.
- Human evaluation datasets hold up to **1,000** prompts, and ground truth is never rated.
- AgentCore online evaluation samples **10%** of sessions by default, with up to **25** evaluators.
- Without gateway tracing, an AgentCore A/B test has no per-variant results.

Sources: Use metrics to understand RAG system performance - Amazon Bedrock (opens in a new tab) · Use metrics to understand model performance - Amazon Bedrock (opens in a new tab) · Create a model evaluation job that uses an LLM as a judge - Amazon Bedrock (opens in a new tab) · Create online evaluation - Amazon Bedrock AgentCore (opens in a new tab) · Run an A/B test for agents hosted outside of AgentCore -

Amazon Bedrock AgentCore (opens in a new tab) · Model evaluation task types in Amazon Bedrock - Amazon Bedrock (opens in a new tab) · Create a human-based model evaluation job - Amazon Bedrock (opens in a new tab) · Starting an automatic model evaluation job in Amazon Bedrock - Amazon Bedrock (opens in a new tab)

Troubleshoot GenAI applications.

Start here

THINK FIRST

When a GenAI app returns errors, drops connections or gives weak answers, how do you find the cause and fix it?

Why it matters: Questions describe an error code, a failed call or a weak answer and ask for its cause or the right fix.

Troubleshooting sorts problems into layers: the API call, the network, the prompt and retrieval. Error codes point to quotas, permissions or capacity, while prompt and retrieval problems show up as weak or invented answers.

Key terms

Term	Meaning
ThrottlingException	HTTP **429**: the request exceeded your account quotas for Amazon Bedrock.
Prompt variant	An alternative prompt configuration, such as a different message, model or inference settings.
Prompt variable	A placeholder in a managed prompt, written with double curly brackets as `{{variable}}`.
Reranking model	A model that reorders retrieved documents by relevance in Retrieve or RetrieveAndGenerate.

Reading Bedrock error codes

The HTTP status code tells you what kind of failure it is. An `overloaded_error` returns **529** for a transient capacity problem. If it carries a `Retry-After` header, wait that long before retrying.

Three Bedrock errors and their causes

Feature	AccessDeniedException	ThrottlingException	ServiceUnavailable
HTTP status	**403**	**429**	**503**
Cause	Not enough permissions for the action	Account quotas exceeded	High demand or temporary capacity limits

EXAM TRAP

A **503** is not a quota problem. Breaching account quotas or rate limits returns a **429** ThrottlingException instead.

Dropped idle connections

Idle connections through NAT or VPC endpoints drop after 350 seconds. NAT Gateways, interface VPC endpoints and Network Load Balancers drop idle TCP connections without telling the client. Turn on TCP keep-alive by setting `tcp_keepalive` to True in the boto3 `Config`. It defaults to False.

Prompt formats per model

Some models expect specific markers in the prompt. Claude prompts sent through the API must contain `\n\nHuman:` and `\n\nAssistant:`. Wrap Claude few-shot demonstrations in example tags. For Titan models, ending the prompt with `\n` helps performance.

EXAM TRAP

Models called through the API do not remember earlier requests. Include the previous interaction in the current prompt to keep context.

Reducing hallucinations

Refine the prompt, add RAG, or try a different model. A default output such as "I don't know" can help prevent plausible-sounding guesses when the model has low confidence. The maximum new tokens setting caps output length. For incomplete answers, check context window overflow, chunking and truncation errors.

Managed prompt syntax

Prompt variables need double curly brackets. Writing `{variable}` instead of `{{variable}}` raises a ValidationException. To run a managed prompt, pass its ARN as `modelId` and its values in `promptVariables`. Messages you add in Converse come after the prompt's own messages.

EXAM TRAP

A Converse call with a managed prompt cannot override its settings. It cannot include `inferenceConfig`, `system`, `toolConfig` or `additionalModelRequestFields`.

Maintaining prompts

Compare variants, then save a version for production. Prompt management lets you test variants and save versions as you iterate. Create version makes a snapshot that production can use. Test values for variables are temporary and are not saved. CloudWatch Logs helps diagnose prompt confusion, and X-Ray supports prompt observability pipelines.

Retrieval problems

Rerank results or split complex queries to retrieve better context. A reranking model reorders documents from Retrieve or RetrieveAndGenerate. `AgenticRetrieveStream` breaks a complex query into sub-queries and checks whether the results are enough. RetrieveAndGenerate answers with citations to specific chunks. Also check embedding quality and drift.

Agents and evaluations

Agent templates and session settings explain many agent surprises. Advanced prompts let you edit templates and add hand-curated few-shot examples. The post-processing template is off by default. Reusing a `sessionId` continues a session only until `idleSessionTTLInSeconds` expires. In AgentCore Evaluation, `LogEventMissingException` means a span lacks its log event.

> **EXAM TRAP**
>
> The AgentCore Evaluation diagnostic assistant queries your CloudWatch log groups to find root causes. If it cannot query them, check credentials, Region and session ID first. The agent must also have run in the last **24** hours.

Key takeaways

- A **429** means account quotas were exceeded; **503** and **529** mean temporary capacity limits.
- Idle connections through NAT Gateways or VPC endpoints drop after **350** seconds, so enable TCP keep-alive.
- Managed prompt variables use `{{variable}}`; single brackets raise a ValidationException.
- `guardrailConfig` guards the whole managed prompt; `guardContent` blocks guard only themselves.
- A SageMaker `model.tar.gz` that contains symlinks makes model creation fail.

Sources: Troubleshooting Amazon Bedrock API Error Codes - Amazon Bedrock (opens in a new tab) · Prompt engineering concepts - Amazon Bedrock (opens in a new tab) · Construct and store reusable prompts with Prompt management in Amazon Bedrock - Amazon Bedrock (opens in a new tab) · Troubleshooting - Amazon Bedrock (opens in a new tab) · Retrieving information from data sources using Amazon Bedrock Knowledge Bases - Amazon Bedrock (opens in a new tab) · Design a prompt - Amazon Bedrock (opens in a new tab) · Exam Guide (AIP-C01) AWS Certified Generative AI Developer - Professional (opens in a new tab) · Test a prompt using Prompt management - Amazon Bedrock (opens in a new tab)

MockCert

Take an AWS Certified Generative AI Developer - Professional mock exam the way the real one feels

5 timed mock exams · 75 questions in 180 minutes each · one is free

- Free to start: study notes, explained quizzes, flashcards and the first mock exam
- Timed mock exams built to the official exam guide
- A verifiable Certificate of Assessment when you pass a premium mock

Explore AWS Certified Generative AI Developer - Professional on...

Open it on your phone

mockcert.com/aws-aip-c01/

Scan the code with your phone's camera.