Aquin Cloud AI BOK Atlas

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

Purpose: preserve the full AWS AIP-C01 requirements map and connect it to professional cloud AI writing, personal Aquin writing, and research artifacts.

Source: AWS official exam guide HTML pages and official AWS-generated PDF, archived offline in this package.

Domain	Weight	Tasks	Skills	Writing emphasis
1. Foundation Model Integration, Data Management, and Compliance	31%	6	28	Context engineering, model selection, FM-ready data, vector stores, retrieval, prompt governance.
2. Implementation and Integration	26%	5	25	Agents, tools, deployment, integration, APIs, development patterns, troubleshooting.
3. AI Safety, Security, and Governance	20%	4	15	Safety, security, privacy, governance, source tracking, transparency, responsible AI.
4. Operational Efficiency and Optimization for GenAI Applications	12%	3	16	Cost, tokens, caching, latency, retrieval performance, observability, FM operations.
5. Testing, Validation, and Troubleshooting	11%	2	14	Evaluation, RAG quality, agent performance, deployment validation, failure troubleshooting.

AIP-C01 Technologies and Concepts

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

Professional blog pillars

• Production GenAI architecture and requirements discipline.
• Context and retrieval engineering: RAG, vector stores, embeddings, metadata, query handling, and GraphRAG-style structure.
• Prompt governance, prompt QA, and prompt lifecycle operations.
• Agentic systems with tools, memory/state, human review, and controlled boundaries.
• Safety, security, privacy, auditability, and responsible AI.
• Optimization: tokens, latency, caching, throughput, cost, and model-routing decisions.
• Evaluation, observability, RAG testing, troubleshooting, deployment validation, and continuous improvement.

Task-to-Writing Map

Task	Official task	Professional cloud post angle	Personal Aquin companion angle	Research artifact
1.1	Analyze requirements and design GenAI solutions	What Production GenAI Actually Requires: architecture, constraints, and proof-of-concept discipline	Aquin first principles: what should a private knowledge machine be responsible for?	Architecture decision records and requirement-to-capability map
1.2	Select and configure FMs	Model selection, model routing, and resilient FM application patterns on AWS	Local model choice, fallback, and graceful degradation for private Apple software	Model-selection matrix and fallback/resilience pattern catalog
1.3	Implement data validation and processing pipelines for FM consumption	Data readiness for foundation-model applications	Private source capture and local preprocessing before anything becomes knowledge	FM-consumption data readiness checklist
1.4	Design and implement vector store solutions	Vector stores are not knowledge management	Local vector indexes as derived artifacts, not canonical knowledge	Vector-store metadata profile and maintenance checklist
1.5	Design retrieval mechanisms for FM augmentation	RAG is a retrieval engineering system	Private GraphRAG on a Mac: from chunks to claim/source/edge context packs	Retrieval evaluation harness and GraphRAG/OKF context-pack schema
1.6	Implement prompt engineering strategies and governance for FM interactions	From prompt engineering to prompt governance	Shortcuts as deterministic automation; local LLM prompts as reviewable proposals	Prompt governance template and regression-test checklist
2.1	Implement agentic AI solutions and tool integrations	Agentic AI needs control planes, tools, memory, and human review gates	Aquin agents should propose, inspect, and route; humans should accept canonical changes	Agent boundary diagram and human-review-gate policy
2.2	Implement model deployment strategies	Deploying foundation-model workloads without pretending they are ordinary web apps	Local and private deployment patterns for model-backed Mac apps	Deployment strategy matrix for FM workloads
2.3	Design and implement enterprise integration architectures	Enterprise integration patterns for GenAI systems	Aquin is not enterprise cloud; it is personal semantic infrastructure with export seams	Cloud/local semantic infrastructure comparison matrix
2.4	Implement FM API integrations	Reliable FM APIs: streaming, routing, retries, and fallbacks	Local APIs, CLI hooks, and deterministic automation boundaries	FM API reliability and model-routing checklist
2.5	Implement application integration patterns and development tools	Developer tooling and application patterns for production GenAI	Mac-native workflow surfaces: files, menus, Shortcuts, CLI, and build logs	Application integration pattern catalog
3.1	Implement input and output safety controls	Guardrails, hallucination controls, and adversarial testing for GenAI	Objections, hallucination checks, and local dispute workflows	Threat model for prompt injection, jailbreaks, unsafe outputs, and hallucinations
3.2	Implement data security and privacy controls	Privacy-preserving GenAI: PII, masking, retention, and secure boundaries	Privacy-by-default knowledge work: local files, selective disclosure, and redaction	Privacy-risk register and sensitive-data handling policy
3.3	Implement AI governance and compliance mechanisms	Source lineage, audit logs, and governance metadata in GenAI systems	Trust passports without the cloud: provenance, review states, and local signatures	Trust-passport schema and provenance/governance mapping
3.4	Implement responsible AI principles	Responsible AI as an engineering discipline	Evidence presentation, uncertainty, and why trust passports are not truth certificates	Evidence/uncertainty/fairness reporting template
4.1	Implement cost optimization and resource efficiency strategies	Token economics: context compression, caching, and cost control	Claude Shannon for personal knowledge: entropy, compression, and redundant evidence	Rate-distortion allocator and token-cost model
4.2	Optimize application performance	Latency, retrieval performance, and model-configuration tradeoffs	Context is a channel: how Aquin should allocate attention and preserve signal	Performance scenario table and retrieval-speed benchmark plan
4.3	Implement monitoring systems for GenAI applications	Observability for GenAI: hallucinations, drift, costs, traces, and tool calls	A local entropy audit for stale claims, missing citations, and broken trails	GenAI observability dashboard specification
5.1	Implement evaluation systems for GenAI	How to evaluate GenAI systems beyond vibes	Aquin evaluation: disputatio, human feedback, retrieval tests, and agent task checks	Evaluation benchmark set and deployment validation checklist
5.2	Troubleshoot GenAI applications	A consultant’s map of RAG, prompt, context, and agent failure modes	Debugging private knowledge systems: prompt drift, bad chunks, and lost context	Failure-mode catalog and troubleshooting decision tree

Complete Skill Catalog

The official AWS PDF is included for exact wording. This appendix preserves the full skill map by ID, task, requirement, and writing bridge.

Domain 1: Foundation Model Integration, Data Management, and Compliance (31%)

Task 1.1: Analyze requirements and design GenAI solutions

Skill	Official requirement	Cloud angle	Aquin / research bridge
1.1.1	Create comprehensive architectural designs that align with specific business needs and technical constraints (for example, by using appropriate FMs, integration patterns, deployment strategies).	What Production GenAI Actually Requires: architecture, constraints, and proof-of-concept discipline	Architecture decision records and requirement-to-capability map
1.1.2	Develop technical proof-of-concept implementations to validate feasibility, performance characteristics, and business value before proceeding to full-scale deployment (for example, by using Amazon Bedrock).	What Production GenAI Actually Requires: architecture, constraints, and proof-of-concept discipline	Architecture decision records and requirement-to-capability map
1.1.3	Create standardized technical components to ensure consistent implementation across multiple deployment scenarios (for example, by using the AWS Well-Architected Framework, AWS WA Tool Generative AI Lens).	What Production GenAI Actually Requires: architecture, constraints, and proof-of-concept discipline	Architecture decision records and requirement-to-capability map

Task 1.2: Select and configure FMs

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

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

Skill	Official requirement	Cloud angle	Aquin / research bridge
1.3.1	Create comprehensive data validation workflows to ensure data meets quality standards for FM consumption (for example, by using AWS Glue Data Quality, SageMaker Data Wrangler, custom Lambda functions, Amazon CloudWatch metrics).	Data readiness for foundation-model applications	FM-consumption data readiness checklist
1.3.2	Create data processing workflows to handle complex data types including text, image, audio, and tabular data with specialized processing requirements for FM consumption (for example, by using Amazon Bedrock multimodal models, SageMaker Processing, AWS Transcribe, advanced multimodal pipeline architectures).	Data readiness for foundation-model applications	FM-consumption data readiness checklist
1.3.3	Format input data for FM inference according to model-specific requirements (for example, by using JSON formatting for Amazon Bedrock API requests, structured data preparation for SageMaker AI endpoints, conversation formatting for dialog-based applications).	Data readiness for foundation-model applications	FM-consumption data readiness checklist
1.3.4	Enhance input data quality to improve FM response quality and consistency (for example, by using Amazon Bedrock to reformat text, Amazon Comprehend to extract entities, Lambda functions to normalize data).	Data readiness for foundation-model applications	FM-consumption data readiness checklist

Task 1.4: Design and implement vector store solutions

Skill	Official requirement	Cloud angle	Aquin / research bridge
1.4.1	Create advanced vector database architectures specifically for FM augmentation to enable efficient semantic retrieval beyond traditional search capabilities (for example, by using Amazon Bedrock Knowledge Bases for hierarchical organization, Amazon OpenSearch Service with the Neural plugin for Amazon Bedrock integration for topic-based segmentation, Amazon RDS with Amazon S3 document repositories, Amazon DynamoDB with vector databases for metadata and embeddings).	Vector stores are not knowledge management	Vector-store metadata profile and maintenance checklist
1.4.2	Develop comprehensive metadata frameworks to improve search precision and context awareness for FM interactions (for example, by using S3 object metadata for document timestamps, custom attributes for authorship information, tagging systems for domain classification).	Vector stores are not knowledge management	Vector-store metadata profile and maintenance checklist
1.4.3	Implement high-performance vector database architectures to optimize semantic search performance at scale for FM retrieval (for example, by using OpenSearch sharding strategies, multi-index approaches for specialized domains, hierarchical indexing techniques).	Vector stores are not knowledge management	Vector-store metadata profile and maintenance checklist
1.4.4	Use AWS services to create integration components to connect with resources (for example, document management systems, knowledge bases, internal wikis for comprehensive data integration in GenAI applications).	Vector stores are not knowledge management	Vector-store metadata profile and maintenance checklist
1.4.5	Design and deploy data maintenance systems to ensure that vector stores contain current and accurate information for FM augmentation (for example, by using incremental update mechanisms, real-time change detection systems, automated synchronization workflows, scheduled refresh pipelines).	Vector stores are not knowledge management	Vector-store metadata profile and maintenance checklist

Task 1.5: Design retrieval mechanisms for FM augmentation

Skill	Official requirement	Cloud angle	Aquin / research bridge
1.5.1	Develop effective document segmentation approaches to optimize retrieval performance for FM context augmentation (for example, by using Amazon Bedrock chunking capabilities, Lambda functions to implement fixed-size chunking, custom processing for hierarchical chunking based on content structure).	RAG is a retrieval engineering system	Retrieval evaluation harness and GraphRAG/OKF context-pack schema
1.5.2	Select and configure optimal embedding solutions to create efficient vector representations for semantic search (for example, by using Amazon Titan embeddings based on dimensionality and domain fit, by evaluating performance characteristics of Amazon Bedrock embedding models, by using Lambda functions to batch generate embeddings).	RAG is a retrieval engineering system	Retrieval evaluation harness and GraphRAG/OKF context-pack schema
1.5.3	Deploy and configure vector search solutions to enable semantic search capabilities for FM augmentation (for example, by using OpenSearch Service with vector search capabilities, Amazon Aurora with the pgvector extension, Amazon Bedrock Knowledge Bases with managed vector store functionality).	RAG is a retrieval engineering system	Retrieval evaluation harness and GraphRAG/OKF context-pack schema
1.5.4	Create advanced search architectures to improve the relevance and accuracy of retrieved information for FM context (for example, by using OpenSearch for semantic search, hybrid search that combines keywords and vectors, Amazon Bedrock reranker models).	RAG is a retrieval engineering system	Retrieval evaluation harness and GraphRAG/OKF context-pack schema
1.5.5	Develop sophisticated query handling systems to improve the retrieval effectiveness and result quality for FM augmentation (for example, by using Amazon Bedrock for query expansion, Lambda functions for query decomposition, Step Functions for query transformation).	RAG is a retrieval engineering system	Retrieval evaluation harness and GraphRAG/OKF context-pack schema
1.5.6	Create consistent access mechanisms to enable seamless integration with FMs (for example, by using function calling interfaces for vector search, Model Context Protocol [MCP] clients for vector queries, standardized API patterns for retrieval augmentation).	RAG is a retrieval engineering system	Retrieval evaluation harness and GraphRAG/OKF context-pack schema

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

Skill	Official requirement	Cloud angle	Aquin / research bridge
1.6.1	Create effective model instruction frameworks to control FM behavior and outputs (for example, by using Amazon Bedrock Prompt Management to enforce role definitions, Amazon Bedrock Guardrails to enforce responsible AI guidelines, template configurations to format responses).	From prompt engineering to prompt governance	Prompt governance template and regression-test checklist
1.6.2	Build interactive AI systems to maintain context and improve user interactions with FMs (for example, by using Step Functions for clarification workflows, Amazon Comprehend for intent recognition, DynamoDB for conversation history storage).	From prompt engineering to prompt governance	Prompt governance template and regression-test checklist
1.6.3	Implement comprehensive prompt management and governance systems to ensure consistency and oversight of FM operations (for example, by using Amazon Bedrock Prompt Management to create parameterized templates and approval workflows, Amazon S3 to store template repositories, AWS CloudTrail to track usage, Amazon CloudWatch Logs to log access).	From prompt engineering to prompt governance	Prompt governance template and regression-test checklist

Skill	Official requirement	Cloud angle	Aquin / research bridge
1.6.4	Develop quality assurance systems to ensure prompt effectiveness and reliability for FMs (for example, by using Lambda functions to verify expected output, Step Functions to test edge cases, CloudWatch to test prompt regression).	From prompt engineering to prompt governance	Prompt governance template and regression-test checklist
1.6.5	Enhance FM performance to refine prompts iteratively and improve response quality beyond basic prompting techniques (for example, by using structured input components, output format specifications, chain-of-thought instruction patterns, feedback loops).	From prompt engineering to prompt governance	Prompt governance template and regression-test checklist
1.6.6	Design complex prompt systems to handle sophisticated tasks with FMs (for example, by using Amazon Bedrock Prompt Flows for sequential prompt chains, conditional branching based on model responses, reusable prompt components, integrated pre-processing and post- processing steps).	From prompt engineering to prompt governance	Prompt governance template and regression-test checklist

Domain 2: Implementation and Integration (26%)

Task 2.1: Implement agentic AI solutions and tool integrations

Skill	Official requirement	Cloud angle	Aquin / research bridge
2.1.1	Develop intelligent autonomous systems with appropriate memory and state management capabilities (for example, by using Strands Agents and AWS Agent Squad for multi- agent systems, MCP for agent-tool interactions).	Agentic AI needs control planes, tools, memory, and human review gates	Agent boundary diagram and human-review-gate policy
2.1.2	Create advanced problem-solving systems to give FMs the ability to break down and solve complex problems by following structured reasoning steps (for example, by using Step Functions to implement ReAct patterns and chain-of-thought reasoning approaches).	Agentic AI needs control planes, tools, memory, and human review gates	Agent boundary diagram and human-review-gate policy
2.1.3	Develop safeguarded AI workflows to ensure controlled FM behavior (for example, by using Step Functions to implement stopping conditions, Lambda functions to implement timeout mechanisms, IAM policies to enforce resource boundaries, circuit breakers to mitigate failures).	Agentic AI needs control planes, tools, memory, and human review gates	Agent boundary diagram and human-review-gate policy
2.1.4	Create sophisticated model coordination systems to optimize performance across multiple capabilities (for example, by using specialized FMs to perform complex tasks, custom aggregation logic for model ensembles, model selection frameworks).	Agentic AI needs control planes, tools, memory, and human review gates	Agent boundary diagram and human-review-gate policy
2.1.5	Develop collaborative AI systems to enhance FM capabilities with human expertise (for example, by using Step Functions to orchestrate review and approval processes, API Gateway to implement feedback collection mechanisms, human augmentation patterns).	Agentic AI needs control planes, tools, memory, and human review gates	Agent boundary diagram and human-review-gate policy
2.1.6	Implement intelligent tool integrations to extend FM capabilities and to ensure reliable tool operations (for example, by using the Strands API to implement custom behaviors, standardized function definitions, Lambda functions to implement error handling and parameter validation).	Agentic AI needs control planes, tools, memory, and human review gates	Agent boundary diagram and human-review-gate policy
2.1.7	Develop model extension frameworks to enhance FM capabilities (for example, by using Lambda functions to implement stateless MCP servers that provide lightweight tool access, Amazon ECS to implement MCP servers that provide complex tools, MCP client libraries to ensure consistent access patterns).	Agentic AI needs control planes, tools, memory, and human review gates	Agent boundary diagram and human-review-gate policy

Task 2.2: Implement model deployment strategies

Skill	Official requirement	Cloud angle	Aquin / research bridge
2.2.1	Deploy FMs based on specific application needs and performance requirements (for example, by using Lambda functions for on-demand invocation, Amazon Bedrock provisioned throughput configurations, SageMaker AI endpoints to implement hybrid solutions).	Deploying foundation-model workloads without pretending they are ordinary web apps	Deployment strategy matrix for FM workloads
2.2.2	Deploy FM solutions by addressing unique challenges of large language models (LLMs) that differ from traditional ML deployments (for example, by implementing container- based deployment patterns that are optimized for memory requirements, GPU utilization, and token processing capacity, by following specialized model loading strategies).	Deploying foundation-model workloads without pretending they are ordinary web apps	Deployment strategy matrix for FM workloads
2.2.3	Develop optimized FM deployment approaches to balance performance and resource requirements for GenAI workloads (for example, by selecting appropriate models, by using smaller pre-trained models for specific tasks, by using API-based model cascading to perform routine queries).	Deploying foundation-model workloads without pretending they are ordinary web apps	Deployment strategy matrix for FM workloads

Task 2.3: Design and implement enterprise integration architectures

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

Task 2.4: Implement FM API integrations

Skill	Official requirement	Cloud angle	Aquin / research bridge
2.4.1	Create flexible model interaction systems (for example, by using Amazon Bedrock APIs to manage synchronous requests from various compute environments, language-specific AWS SDKs and Amazon SQS for asynchronous processing, API Gateway to provide custom API clients with request validation).	Reliable FM APIs: streaming, routing, retries, and fallbacks	FM API reliability and model-routing checklist
2.4.2	Develop real-time AI interaction systems to provide immediate feedback from FM (for example, by using Amazon Bedrock streaming APIs for incremental response delivery, WebSockets or server-sent events to generate text in real time, API Gateway to implement chunked transfer encoding).	Reliable FM APIs: streaming, routing, retries, and fallbacks	FM API reliability and model-routing checklist
2.4.3	Create resilient FM systems to ensure reliable operations (for example, by using the AWS SDK for exponential backoff, API Gateway to manage rate limiting, fallback mechanisms for graceful degradation, AWS X-Ray to provide observability across service boundaries).	Reliable FM APIs: streaming, routing, retries, and fallbacks	FM API reliability and model-routing checklist
2.4.4	Develop intelligent model routing systems to optimize model selection (for example, by using application code to implement static routing configurations, Step Functions for dynamic content-based routing to specialized FMs, intelligent model routing based on metrics, API Gateway with request transformations for routing logic).	Reliable FM APIs: streaming, routing, retries, and fallbacks	FM API reliability and model-routing checklist

Task 2.5: Implement application integration patterns and development tools

Skill	Official requirement	Cloud angle	Aquin / research bridge
2.5.1	Create FM API interfaces to address the specific requirements of GenAI workloads (for example, by using API Gateway to handle streaming responses, token limit management, retry strategies to handle model timeouts).	Developer tooling and application patterns for production GenAI	Application integration pattern catalog
2.5.2	Develop accessible AI interfaces to accelerate adoption and integration of FMs (for example, by using AWS Amplify to develop declarative UI components, OpenAPI specifications for API-first development approaches, Amazon Bedrock Prompt Flows for no-code workflow builders).	Developer tooling and application patterns for production GenAI	Application integration pattern catalog
2.5.3	Create business system enhancements (for example, by using Lambda functions to implement customer relationship management [CRM] enhancements, Step Functions to orchestrate document processing systems, Amazon Bedrock Data Automation to manage automated data processing workflows).	Developer tooling and application patterns for production GenAI	Application integration pattern catalog
2.5.4	Enhance developer productivity to accelerate development workflows for GenAI applications (for example, by using Amazon Q Developer to generate and refactor code, code suggestions for API assistance, AI component testing, performance optimization).	Developer tooling and application patterns for production GenAI	Application integration pattern catalog

Skill	Official requirement	Cloud angle	Aquin / research bridge
2.5.5	Develop advanced GenAI applications to implement sophisticated AI capabilities (for example, by using Strands Agents and AWS Agent Squad for AWS native orchestration, Step Functions to orchestrate agent design patterns, Amazon Bedrock to manage prompt chaining patterns).	Developer tooling and application patterns for production GenAI	Application integration pattern catalog
2.5.6	Improve troubleshooting efficiency for FM applications (for example, by using CloudWatch Logs Insights to analyze prompts and responses, X-Ray to trace FM API calls, Amazon Q Developer to implement GenAI-specific error pattern recognition).	Developer tooling and application patterns for production GenAI	Application integration pattern catalog

Domain 3: AI Safety, Security, and Governance (20%)

Task 3.1: Implement input and output safety controls

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

Task 3.2: Implement data security and privacy controls

Skill	Official requirement	Cloud angle	Aquin / research bridge
3.2.1	Develop protected AI environments to ensure comprehensive security for FM deployments (for example, by using VPC endpoints to isolate networks, IAM policies to enforce secure data access patterns, AWS Lake Formation to provide granular data access, CloudWatch to monitor data access).	Privacy-preserving GenAI: PII, masking, retention, and secure boundaries	Privacy-risk register and sensitive-data handling policy
3.2.2	Develop privacy-preserving systems to protect sensitive information during FM interactions (for example, by using Amazon Comprehend and Amazon Macie to detect personally identifiable information [PII], Amazon Bedrock native data privacy features, Amazon Bedrock guardrails to filter outputs, Amazon S3 Lifecycle configurations to implement data retention policies).	Privacy-preserving GenAI: PII, masking, retention, and secure boundaries	Privacy-risk register and sensitive-data handling policy
3.2.3	Create privacy-focused AI systems to protect user privacy while maintaining FM utility and effectiveness (for example, by using data masking techniques, Amazon Comprehend PII detection, anonymization strategies for sensitive information, Amazon Bedrock guardrails).	Privacy-preserving GenAI: PII, masking, retention, and secure boundaries	Privacy-risk register and sensitive-data handling policy

Task 3.3: Implement AI governance and compliance mechanisms

Skill	Official requirement	Cloud angle	Aquin / research bridge
3.3.1	Develop compliance frameworks to ensure regulatory compliance for FM deployments (for example, by using SageMaker AI to develop programmatic model cards, AWS Glue to automatically track data lineage, metadata tagging for systematic data source attribution, CloudWatch Logs to collect comprehensive decision logs).	Source lineage, audit logs, and governance metadata in GenAI systems	Trust-passport schema and provenance/governance mapping

Skill	Official requirement	Cloud angle	Aquin / research bridge
3.3.2	Implement data source tracking to maintain traceability in GenAI applications (for example, by using AWS Glue Data Catalog to register data sources, metadata tagging for source attribution in FM-generated content, CloudTrail for audit logging).	Source lineage, audit logs, and governance metadata in GenAI systems	Trust-passport schema and provenance/governance mapping
3.3.3	Create organizational governance systems to ensure consistent oversight of FM implementations (for example, by using comprehensive frameworks that align with organizational policies, regulatory requirements, and responsible AI principles).	Source lineage, audit logs, and governance metadata in GenAI systems	Trust-passport schema and provenance/governance mapping
3.3.4	Implement continuous monitoring and advanced governance controls to support safety audits and regulatory readiness (for example, by using automated detection for misuse, drift, and policy violations, bias drift monitoring, automated alerting and remediation workflows, token-level redaction, response logging, AI output policy filters).	Source lineage, audit logs, and governance metadata in GenAI systems	Trust-passport schema and provenance/governance mapping

Task 3.4: Implement responsible AI principles

Skill	Official requirement	Cloud angle	Aquin / research bridge
3.4.1	Develop transparent AI systems in FM outputs (for example, by using reasoning displays to provide user-facing explanations, CloudWatch to collect confidence metrics and quantify uncertainty, evidence presentation for source attribution, Amazon Bedrock agent tracing to provide reasoning traces).	Responsible AI as an engineering discipline	Evidence/uncertainty/fairness reporting template
3.4.2	Apply fairness evaluations to ensure unbiased FM outputs (for example, by using pre- defined fairness metrics in CloudWatch, Amazon Bedrock Prompt Management and Amazon Bedrock Prompt Flows to perform systematic A/B testing, Amazon Bedrock with LLM-as-a-judge solutions to perform automated model evaluations).	Responsible AI as an engineering discipline	Evidence/uncertainty/fairness reporting template
3.4.3	Develop policy-compliant AI systems to ensure adherence to responsible AI practices (for example, by using Amazon Bedrock guardrails based on policy requirements, model cards to document FM limitations, Lambda functions to perform automated compliance checks).	Responsible AI as an engineering discipline	Evidence/uncertainty/fairness reporting template

Domain 4: Operational Efficiency and Optimization for GenAI Applications (12%)

Task 4.1: Implement cost optimization and resource efficiency strategies

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

Task 4.2: Optimize application performance

Skill	Official requirement	Cloud angle	Aquin / research bridge
4.2.1	Create responsive AI systems to address latency-cost tradeoffs and improve the user experience with FMs (for example, by using pre-computation to perform predictable queries, latency-optimized Amazon Bedrock models for time-sensitive applications, parallel requests for complex workflows, response streaming, performance benchmarking).	Latency, retrieval performance, and model-configuration tradeoffs	Performance scenario table and retrieval-speed benchmark plan

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

Task 4.3: Implement monitoring systems for GenAI applications

Skill	Official requirement	Cloud angle	Aquin / research bridge
4.3.1	Create holistic observability systems to provide complete visibility into FM application performance (for example, by using operational metrics, performance tracing, FM interaction tracing, business impact metrics with custom dashboards).	Observability for GenAI: hallucinations, drift, costs, traces, and tool calls	GenAI observability dashboard specification
4.3.2	Implement comprehensive GenAI monitoring systems to proactively identify issues and evaluate key performance indicators specific to FM implementations (for example, by using CloudWatch to track token usage; prompt effectiveness; hallucination rates; and response quality, anomaly detection for token burst patterns and response drift, Amazon Bedrock Model Invocation Logs to perform detailed request and response analysis, performance benchmarks, cost anomaly detection).	Observability for GenAI: hallucinations, drift, costs, traces, and tool calls	GenAI observability dashboard specification
4.3.3	Develop integrated observability solutions to provide actionable insights for FM applications (for example, by using operational metric dashboards, business impact visualizations, compliance monitoring, forensic traceability and audit logging, user interaction tracking, model behavior pattern tracking).	Observability for GenAI: hallucinations, drift, costs, traces, and tool calls	GenAI observability dashboard specification
4.3.4	Create tool performance frameworks to ensure optimal tool operation and utilization for FMs (for example, by using call pattern tracking, performance metric collection, tool calling observability and multi-agent coordination tracking, usage baselines for anomaly detection).	Observability for GenAI: hallucinations, drift, costs, traces, and tool calls	GenAI observability dashboard specification
4.3.5	Create vector store operational management systems to ensure optimal vector store operation and reliability for FM augmentation (for example, by using performance monitoring for vector databases, automated index optimization routines, data quality validation processes).	Observability for GenAI: hallucinations, drift, costs, traces, and tool calls	GenAI observability dashboard specification
4.3.6	Develop FM-specific troubleshooting frameworks to identify unique GenAI failure modes that are not present in traditional ML systems (for example, by using golden datasets to detect hallucinations, output diffing techniques to conduct response consistency analysis, reasoning path tracing to identify logical errors, specialized observability pipelines).	Observability for GenAI: hallucinations, drift, costs, traces, and tool calls	GenAI observability dashboard specification

Domain 5: Testing, Validation, and Troubleshooting (11%)

Task 5.1: Implement evaluation systems for GenAI

Skill	Official requirement	Cloud angle	Aquin / research bridge
5.1.1	Develop comprehensive assessment frameworks to evaluate the quality and effectiveness of FM outputs beyond traditional ML evaluation approaches (for example, by using metrics for relevance, factual accuracy, consistency, and fluency).	How to evaluate GenAI systems beyond vibes	Evaluation benchmark set and deployment validation checklist

Skill	Official requirement	Cloud angle	Aquin / research bridge
5.1.2	Create systematic model evaluation systems to identify optimal configurations (for example, by using Amazon Bedrock Model Evaluations, A/B testing and canary testing of FMs, multi-model evaluation, cost-performance analysis to measure token efficiency, latency-to- quality ratios, and business outcomes).	How to evaluate GenAI systems beyond vibes	Evaluation benchmark set and deployment validation checklist
5.1.3	Develop user-centered evaluation mechanisms to continuously improve FM performance based on user experience (for example, by using feedback interfaces, rating systems for model outputs, annotation workflows to assess response quality).	How to evaluate GenAI systems beyond vibes	Evaluation benchmark set and deployment validation checklist
5.1.4	Create systematic quality assurance processes to maintain consistent performance standards for FMs (for example, by using continuous evaluation workflows, regression testing for model outputs, automated quality gates for deployments).	How to evaluate GenAI systems beyond vibes	Evaluation benchmark set and deployment validation checklist
5.1.5	Develop comprehensive assessment systems to ensure thorough evaluation from multiple perspectives for FM outputs (for example, by using RAG evaluation, automated quality assessment with LLM-as-a-Judge techniques, human feedback collection interfaces).	How to evaluate GenAI systems beyond vibes	Evaluation benchmark set and deployment validation checklist
5.1.6	Implement retrieval quality testing to evaluate and optimize information retrieval components for FM augmentation (for example, by using relevance scoring, context matching verification, retrieval latency measurements).	How to evaluate GenAI systems beyond vibes	Evaluation benchmark set and deployment validation checklist
5.1.7	Develop agent performance frameworks to ensure that agents perform tasks correctly and efficiently (for example, by using task completion rate measurements, tool usage effectiveness evaluations, Amazon Bedrock Agent evaluations, reasoning quality assessment in multi-step workflows).	How to evaluate GenAI systems beyond vibes	Evaluation benchmark set and deployment validation checklist
5.1.8	Create comprehensive reporting systems to communicate performance metrics and insights effectively to stakeholders for FM implementations (for example, by using visualization tools, automated reporting mechanisms, model comparison visualizations).	How to evaluate GenAI systems beyond vibes	Evaluation benchmark set and deployment validation checklist
5.1.9	Create deployment validation systems to maintain reliability during FM updates (for example, by using synthetic user workflows, AI-specific output validation for hallucination rates and semantic drift, automated quality checks to ensure response consistency).	How to evaluate GenAI systems beyond vibes	Evaluation benchmark set and deployment validation checklist

Task 5.2: Troubleshoot GenAI applications

Skill	Official requirement	Cloud angle	Aquin / research bridge
5.2.1	Resolve content handling issues to ensure that necessary information is processed completely in FM interactions (for example, by using context window overflow diagnostics, dynamic chunking strategies, prompt design optimization, truncation-related error analysis).	A consultant’s map of RAG, prompt, context, and agent failure modes	Failure-mode catalog and troubleshooting decision tree
5.2.2	Diagnose and resolve FM integration issues to identify and fix API integration problems specific to GenAI services (for example, by using error logging, request validation, response analysis).	A consultant’s map of RAG, prompt, context, and agent failure modes	Failure-mode catalog and troubleshooting decision tree
5.2.3	Troubleshoot prompt engineering problems to improve FM response quality and consistency beyond basic prompt adjustments (for example, by using prompt testing frameworks, version comparison, systematic refinement).	A consultant’s map of RAG, prompt, context, and agent failure modes	Failure-mode catalog and troubleshooting decision tree
5.2.4	Troubleshoot retrieval system issues to identify and resolve problems that affect information retrieval effectiveness for FM augmentation (for example, by using model response relevance analysis, embedding quality diagnostics, drift monitoring, vectorization issue resolution, chunking and preprocessing remediation, vector search performance optimization).	A consultant’s map of RAG, prompt, context, and agent failure modes	Failure-mode catalog and troubleshooting decision tree
5.2.5	Troubleshoot prompt maintenance issues to continuously improve the performance of FM interactions (for example, by using template testing and CloudWatch Logs to diagnose prompt confusion, X-Ray to implement prompt observability pipelines, schema validation to detect format inconsistencies, systematic prompt refinement workflows).	A consultant’s map of RAG, prompt, context, and agent failure modes	Failure-mode catalog and troubleshooting decision tree

In-Scope Services

Analytics

Amazon Athena, Amazon EMR, AWS Glue, Amazon Kinesis, Amazon OpenSearch Service, Amazon Quick Sight, Amazon Managed Streaming for Apache Kafka (Amazon MSK)

Application Integration

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

Compute

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

Containers

Amazon ECR, Amazon ECS, Amazon EKS, AWS Fargate

Customer Engagement

Amazon Connect

Database

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

Developer Tools

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

Machine Learning

Amazon Augmented AI, Amazon Bedrock, Amazon Bedrock AgentCore, Amazon Bedrock Knowledge Bases, Amazon Bedrock Prompt Management, Amazon Bedrock Prompt Flows, Amazon Comprehend, Amazon Kendra, Amazon Lex, Amazon Q Business, Amazon Q Business Apps, Amazon Q Developer, Amazon Quick, Amazon Rekognition, Amazon SageMaker AI, Amazon SageMaker Clarify, Amazon SageMaker Data Wrangler, Amazon SageMaker Ground Truth, Amazon SageMaker JumpStart, Amazon SageMaker Model Monitor, Amazon SageMaker Model Registry, Amazon SageMaker Neo, Amazon SageMaker Processing, Amazon SageMaker Unified Studio, Amazon Textract, Amazon Titan, Amazon Transcribe

Management and Governance

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

Migration and Transfer

AWS DataSync, AWS Transfer Family

Networking and Content Delivery

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

Security, Identity, and Compliance

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

Storage

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

Out-of-Scope Services

Important research note: Amazon Managed Blockchain is out of scope for AIP-C01, so Ethereum belongs in the Aquin trust-passport research comparison rather than in the certification core.

Application Integration

Amazon MQ

Analytics

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

Blockchain

Amazon Managed Blockchain (AMB)

Business Applications

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

Cloud Financial Management

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

Compute

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

Containers

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

Customer Engagement

Amazon SES

Database

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

Developer Tools

AWS Cloud9, AWS CloudShell, Amazon CodeGuru, AWS CodeStar, Amazon Corretto

End User Computing

Amazon AppStream 2.0, Amazon WorkLink, Amazon WorkSpaces, Amazon WorkSpaces Web

Frontend Web and Mobile

AWS Device Farm, Amazon Location Service, Amazon Pinpoint

Game Development

Amazon GameLift, Amazon Lumberyard

Internet of Things (IoT)

AWS IoT 1-Click, AWS IoT Analytics, AWS IoT Button, AWS IoT Core, AWS IoT Device Defender, AWS IoT Device Management, AWS IoT Events, AWS IoT FleetWise, AWS IoT Greengrass, AWS IoT SiteWise, AWS IoT TwinMaker

Management and Governance

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

Machine Learning

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

Media Services

Amazon Elastic Transcoder, AWS Elemental MediaConnect, AWS Elemental MediaConvert, AWS Elemental MediaLive, AWS Elemental MediaPackage, AWS Elemental MediaStore, AWS Elemental MediaTailor, Amazon Interactive Video Service, Amazon Kinesis Video Streams, Amazon Nimble Studio

Migration and Transfer

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

Networking and Content Delivery

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

Quantum Technologies

Amazon Braket

Robotics

AWS RoboMaker

Satellite

AWS Ground Station