Qazi Adnan

Srinagar, Jammu and Kashmir
qazi.adnan.2k1@gmail.com — 6006758687

Professional Summary

- AI Engineer with advanced expertise in Machine Learning, Deep Learning, and Generative AI, specializing in LLMs, RAG, and MLOps for enterprise applications on AWS.
- Expert in Amazon SageMaker (Training, Pipelines, Feature Store, Model Registry, Serverless/Multi-Model Endpoints) and Amazon Bedrock (Guardrails, Knowledge Bases, Agents).
- Skilled in deploying microservices with FastAPI and Flask, using EKS/ECS, Step Functions, EventBridge, and Lambda for scalable and observable AI platforms.
- Experienced in vector search, semantic retrieval, and evaluation using OpenSearch, FAISS, pgvector, and hybrid retrieval with reranking strategies.
- Strong in data engineering with S3 data lakes, Glue, Lake Formation, Athena, Redshift, and secure architectures using IAM, VPC, KMS, WAF, and CloudTrail.

Education

National Institute of Technology (NIT) Srinagar B.Tech in Computer Science & Engineering

Experience

Elyspace June 2024 – Present
AI Engineer

- Architected a production-grade GenAI platform on AWS using Amazon Bedrock Agents and Guardrails with OpenSearch vector indices and hybrid search.
- Built end-to-end MLOps pipelines in SageMaker: data prep, HPO, distributed training, registry approval workflows, and CI/CD with CodePipeline.
- Designed low-latency inference with SageMaker Multi-Model Endpoints and Serverless Inference, integrating blue/green and canary deployments.
- Implemented RAG pipelines with Bedrock Knowledge Bases, embeddings in OpenSearch/pgvector, reranking, and prompt safety filtering.
- Established observability with CloudWatch, Model Monitor, Clarify, and OpenTelemetry and defined SLOs for latency, throughput, and accuracy.
- Delivered secure data lakes with S3, Glue, and Lake Formation, fine-grained IAM, encrypted pipelines, and Athena/Redshift analytics.

Skills

Programming:	Python, FastAPI, Flask, SQL, Bash, TypeScript, Node.js
ML/DL:	PyTorch, TensorFlow, scikit-learn, XGBoost, ONNX Runtime, Triton, TensorRT
LLM/GenAI:	Amazon Bedrock, LangChain, LlamaIndex, RAG, Prompt Engineering, Eval Frameworks
AWS AI/ML:	SageMaker (Pipelines, HPO, Feature Store, Clarify, Wrangler, MME, Serverless), Bedrock (Guardrails, KBs, Agents), Kendra, OpenSearch, Comprehend, Textract
Data/Streaming:	S3, Glue, Lake Formation, Athena, Redshift, EMR, MSK Kafka, Kinesis
Infra/Backend:	Lambda, Step Functions, ECS, EKS, API Gateway, SQS, SNS, EventBridge
Security/Ops:	IAM, KMS, Secrets Manager, VPC, PrivateLink, WAF, CloudWatch, CloudTrail
DevOps/IaC:	Docker, Kubernetes, Terraform, AWS CDK, GitHub Actions, CodePipeline
Databases/Search:	PostgreSQL, MySQL, MongoDB, Redis, OpenSearch Vector, pgvector, FAISS
Platforms:	Azure AI Foundry, Vertex AI, Git, Jira, Postman, Swagger

Interests

Applied ML and trustworthy AI, retrieval research, vector search, LLM safety and evaluation, scalable inference systems, and developer platforms for AI.

Enterprise Retrieval-Augmented Generation (RAG) Platform

Overview: A production RAG platform that integrates enterprise knowledge bases with vector search and LLM orchestration to provide accurate, context-aware question answering, summarization, and assistant workflows for internal knowledge and customer-facing applications.

Responsibilities and Features:

- Designed ingestion pipelines to convert heterogeneous sources (Confluence, SharePoint, PDFs, relational DB exports, HTML, and S3 documents) into normalized chunks with metadata and lineage.
- Built embedding pipelines using configurable models, stored vectors in OpenSearch and pgvector for hybrid retrieval, and implemented incremental reindexing and TTL policies.
- Implemented hybrid retrieval combining lexical (BM25) and dense vector scoring with reranking using cross-encoders to boost precision on short queries and high-precision enterprise tasks.
- Integrated Amazon Bedrock Knowledge Bases and a retrieval layer to perform context injection for LLM prompts; added prompt templates, dynamic few-shot selection, and context-aware safety filters.
- Added multi-tenant isolation with per-tenant indices, KMS-encrypted artifacts, IAM roles, and data access policies implemented with Lake Formation and fine-grained OpenSearch role mappings.

Architecture and Components:

- Data Ingestion: S3 staged raw data, Lambda/Glue jobs for parsing, Textract for scanned docs, and Step Functions for orchestrating extraction and chunking.
- Embeddings Storage: Batch and streaming embedding jobs in SageMaker Processing; OpenSearch vector indices + pgvector for transactional retrieval; FAISS for offline experiments.
- Retrieval: Hybrid retriever service (FastAPI) exposing federated search combining BM25 and vector scores, with async reranking using a cross-encoder deployed on SageMaker.
- Orchestration: Bedrock agent for tool calls, knowledge base lookup, and guardrail enforcement; request-level observability via OpenTelemetry and CloudWatch.
- Monitoring: Model drift detection with SageMaker Model Monitor, query-level feedback loop for evaluation, and eval pipelines generating precision/recall and F1 dashboards.

Tech Stack: Python, FastAPI, SageMaker (Processing, Training, Endpoints, Pipelines), Amazon Bedrock, OpenSearch, pgvector, FAISS, Textract, Glue, Step Functions, S3, KMS, IAM, OpenTelemetry.

Key Outcomes:

- Delivered consistent, explainable answers to internal knowledge queries with configurable context budgets and fallbacks to safe responses.
- Enabled fine-grained access and auditability for enterprise knowledge assets and simplified integration for downstream products via a REST/GraphQL facade.

Recommendation System for Job Portal and E-commerce App

Overview: An end-to-end recommendation system architecture supporting two domains: a job marketplace and an e-commerce platform. Designed for relevance, personalization, diversity, and production-grade retraining and A/B experimentation.

Responsibilities and Features:

- Built unified feature stores and real-time feature pipelines to support both session-based and long-term personalization using SageMaker Feature Store and streaming ingestion (Kinesis/MSK).
- Implemented candidate generation for both domains: job matching using semantic embeddings of job descriptions and resumes, and e-commerce candidate lists using collaborative and content-based signals.
- Developed ranking models using gradient-boosted trees (XGBoost) and deep learning (SASRec-style for sessions, and two-tower deep retrieval) with pairwise and pointwise losses.
- Designed business rules layer for freshness, recency, and blacklisting; added contextual bandit exploration for controlled discovery and diversity.
- Set up offline evaluation (holdout, time-split), online A/B testing, and automated retraining pipelines with validation gates and CI/CD deployment to SageMaker Endpoints.

Architecture and Components:

- Data Layer: Event collectors (Kafka/Kinesis), batch ETL with Glue, S3 feature lake, and Feature Store for low-latency access.
- Candidate Generation: Two-tower embedding pipelines, session-based samplers, and metadata filters. Resumes and job descriptions embedded and matched via FAISS/OpenSearch vectors.
- Ranking: Ensemble of GBDT and neural ranking with post-processing for business constraints. Models deployed as scalable SageMaker endpoints with autoscaling and warm pools.
- Serving: Real-time inference API (FastAPI) that composes candidates, calls ranking endpoints, applies

business filters, and returns final ranked lists with explainability metadata.
- Experimentation Monitoring: A/B testing harness, online metrics aggregation (CTR, apply-rate, conversion), model performance monitoring, and drift alerts.

Tech Stack: Python, SageMaker (Training, Endpoints, Pipelines), Feature Store, FAISS, OpenSearch, XGBoost, PyTorch, Kafka/MSK, Kinesis, Glue, S3, Terraform, Prometheus, Grafana.
Key Outcomes:
- Achieved robust candidate recall with personalized ranking while maintaining low end-to-end latency through cached candidate pools and efficient batching.
- Enabled rapid iteration using reproducible pipelines and automatic validation gates for model promotion to production.

LLM Inference Mesh

Overview: A horizontally scalable inference mesh that consolidates multiple LLMs and model variants under a unified serving layer to optimize GPU utilization, reduce cold-starts, and enable cost-effective multi-tenant inference.
Responsibilities and Features:
- Implemented SageMaker Multi-Model Endpoints (MME) and a model router to load/unload models dynamically and route requests by tenant, model version, and SLA.
- Integrated Triton Inference Server for optimized batched GPU inference and ONNX/TorchScript model serving to maximize throughput and minimize latency.
- Built autoscaling policies and warm-inference pools, implemented request batching and micro-batching strategies, and designed a cost-aware scheduling layer to colocate compatible models on the same GPU host.
- Created a model lifecycle manager for rolling updates, blue/green deployments, canary testing, and model registry hooks to automate safe rollouts.
- Exposed a unified API with token-level streaming support, request tracing, and per-request guardrails for prompt safety and cost containment.

Architecture and Components:
- Serving Fabric: MME on SageMaker + Triton for high-performance inference, with an orchestrator that manages model placement and batching.
- Router Gateway: FastAPI gateway that performs authentication, tenant routing, prompt preprocessing, and request deduplication before dispatch to the mesh.
- Observability: End-to-end tracing with OpenTelemetry, latency heatmaps, GPU utilization dashboards, and model-level telemetry for throughput, error rates, and token-cost metrics.
- Cost Optimization: Dynamic model packing, multi-tenant quotas, pre-warming strategies, and usage-driven autoscaling.

Tech Stack: SageMaker MME, Triton Inference Server, Docker, Kubernetes/EKS, FastAPI, ONNX Runtime, PyTorch, Prometheus, OpenTelemetry, CloudWatch.
Key Outcomes:
- Enabled predictable latency and throughput for mixed LLM workloads with improved GPU utilization and operational controls for cost and SLA management.
- Provided infrastructure for rapid model experimentation and safe rollouts across tenants without service disruption.

Document AI Pipeline

Overview: A serverless document processing pipeline that extracts structured information from large sets of documents, performs summarization and QA, and enforces compliance guardrails for sensitive content.
Responsibilities and Features:
- Built an orchestrated pipeline using S3, Step Functions, Textract for OCR, and parallelized extraction jobs to process high-volume document batches.
- Implemented text cleaning, chunking, and semantic embedding workflows to populate knowledge bases for downstream RAG use cases.
- Created summarization and Q&A microservices leveraging Bedrock and SageMaker endpoints with prompt templates, chunk-prioritization heuristics, and answer provenance metadata.
- Added compliance and PII detection layers using Comprehend and custom classifiers, redaction policies, and role-based access to outputs.
- Implemented retry semantics, idempotent processing, and audit trails for regulatory requirements.

Architecture and Components:
- Ingestion: S3 upload triggers Step Functions that orchestrate Textract, validation, and downstream processing.

- Processing: Parallel SageMaker Processing jobs for NLP preprocessing, embedding generation, and metadata extraction.
- Serving: Bedrock-backed summarization and QA endpoints with context-handling, answer justification, and confidence scores.
- Governance: PII detectors, redaction tools, and audit logs stored in encrypted S3 with Lake Formation governance.

Tech Stack: Textract, SageMaker, Amazon Bedrock, Comprehend, Step Functions, S3, Glue, Lambda, OpenSearch, KMS.
Key Outcomes:

- Delivered scalable, auditable document workflows for compliance-sensitive workflows with integrated summarization and evidence trails for generated answers.

Customer Support Agentic AI

Overview: An agentic AI system designed to augment customer support by orchestrating tools, knowledge retrieval, external system actions, and safe automation to resolve tickets, escalate when necessary, and provide summarized case histories.
Responsibilities and Features:

- Designed multi-step agent workflows that combine retrieval (RAG), tool invocation (ticketing system API, CRM calls, knowledge lookup), and action execution under governed policies.
- Built a toolset layer with connectors for Zendesk/Jira/ServiceNow, internal CRM, order systems, and diagnostic utilities; implemented idempotency and authorization checks for action tools.
- Implemented a meta-controller agent that reasons about task decomposition, fallback strategies, and when to involve human agents; included human-in-the-loop escalation triggers and approvals.
- Added hallucination mitigation strategies: provenance attachment for all responses, verification sub-steps (cross-checking data with authoritative APIs), and rollback procedures for automated actions.
- Created conversation memory stores with vectorized summaries, long-term case histories, and token budget management to ensure concise, relevant context for each agent decision.

Architecture and Components:

- Agent Engine: Bedrock agents coordinating tools, with a planner-executor split to generate plans and verify outcomes before committing actions.
- Connectors: Secure, audited connectors to external systems with scoped credentials, fine-grained permissions, and action validation.
- Safety Governance: Guardrails for allowed actions, human approval gates, rate limiting, and audit logs for traceability.
- Observability: Action-level telemetry, success/failure tracking, and user satisfaction signals integrated into retraining pipelines.

Tech Stack: Amazon Bedrock Agents, SageMaker, FastAPI, OAuth2 for connectors, OpenSearch, pgvector, S3, KMS, Terraform, Prometheus.
Key Outcomes:

- Reduced simple-ticket handling time by enabling safe automated resolutions and provided agents with condensed case summaries for faster escalations.
- Increased traceability and confidence through provenance-first responses and enforced verification steps for all automated actions.