CollectIQ — DevOps Project Specification

Purpose: Define AWS infrastructure, Terraform module layout, IAM/security, CI/CD, observability, and runbooks for a system composed of AWS Amplify Hosting (Next.js UI), API Gateway + Lambda backend, and a Multi‑Agent Orchestration pipeline where Amazon Rekognition runs BEFORE the Bedrock reasoning stage. DevOps owns all IaC, deployments, environments, quotas, and cross‑service integrations.

# 1. Architecture Overview

Frontend: Next.js 14 hosted on AWS Amplify Hosting (SSR/ISR). Backend: Lambda functions behind Amazon API Gateway (HTTP API) secured by Cognito JWT authorizer. Orchestration: AWS Step Functions coordinates agents; Amazon EventBridge emits domain events. Computer vision signals are extracted with Amazon Rekognition BEFORE invoking Amazon Bedrock (Supervisor/Sub‑agents) to produce valuation and authenticity reasoning.

High‑level flow:  
1) User uploads image (S3 presigned PUT) → metadata in DynamoDB.  
2) API request (revalue/analyze) hits API Gateway → Lambda → Step Functions.  
3) Step Functions first invokes Rekognition task(s) to extract visual/ocr features.  
4) Outputs (features) are passed to Bedrock Supervisor/Sub‑agents for reasoning (valuation + authenticity synthesis).  
5) Parallel branches (pricing ingestion/fusion, authenticity reasoning) run and then merge; results persisted to DynamoDB; EventBridge emits updates.

# 2. Terraform Layout

infra/terraform/  
 modules/  
 amplify\_hosting/ # Next.js app, env vars, custom domain  
 api\_gateway\_http/ # Routes, JWT authorizer, logging, throttling  
 cognito\_user\_pool/ # User pool, app client, domain, MFA  
 s3\_uploads/ # Private upload bucket + CORS  
 dynamodb\_collectiq/ # Single-table with GSIs  
 lambda\_fn/ # Generic Lambda module (artifact, env, IAM)  
 step\_functions/ # State machines (multi‑agent)  
 eventbridge\_bus/ # Domain event bus + rules + DLQs  
 rekognition\_access/ # IAM policies/endpoints/quotas notes  
 bedrock\_access/ # Bedrock model permissions & quotas  
 cloudwatch\_dashboards/ # KPIs  
 xray/ # Tracing  
 ssm\_secrets/ # External API keys  
 envs/  
 dev/  
 prod/

• Remote state: S3 versioned bucket + DynamoDB state lock per env.

• Resource tagging: {Project=CollectIQ, Env, Owner}.

• CI enforces terraform fmt/validate/plan; approval required for apply; tflint/checkov gate merges.

# 3. Multi‑Agent Orchestration (Step Functions) — Rekognition Before Bedrock

State machine pattern:  
• Input: { userId, cardId, s3Keys{front,back?}, requestId } from API Lambda.  
• Task 1 (RekognitionExtract): Lambda invoking Amazon Rekognition (DetectText, DetectLabels/Custom, feature extraction). Outputs a normalized FeatureEnvelope { ocr, borders, holoVariance, fontMetrics, qualitySignals }.  
• Parallel Branches (after features exist):  
 A) PricingAgent: fetch live comps (eBay, TCGPlayer, PriceCharting), normalize+fuse, compute low/median/high, confidence.  
 B) AuthenticityAgent: Bedrock prompts consume FeatureEnvelope + metadata to produce authenticityScore ∈ [0,1] + rationale.  
• Join (Aggregator): persists merged result into DynamoDB and publishes EventBridge events.  
• Error handling: retries w/ backoff per task; catch to DLQ (SQS) with correlation IDs; partial results allowed in Aggregator.

Terraform deliverables:  
• step\_functions module to accept ASL (Amazon States Language) JSON.  
• IAM roles for each task Lambda with least‑privilege (Rekognition, Bedrock, S3, DDB, Secrets).  
• EventBridge rules for CardValuationUpdated/AuthenticityFlagged with DLQs.

# 4. Amazon Bedrock Integration (Supervisor + Sub‑Agents)

DevOps tasks:  
• Enable Bedrock in region; allowlisted foundation model(s) as required.  
• IAM policies for Lambdas to invoke Bedrock Runtime (bedrock:InvokeModel / InvokeModelWithResponseStream).  
• Quotas/limits monitoring; configurable concurrency/timeout for Bedrock tasks via Lambda env.  
• Optional VPC endpoints if private networking is required (advanced).

Runtime contract:  
• Supervisor prompt receives FeatureEnvelope + card metadata; routes to sub‑agents (valuation, authenticity) via prompt‑based or programmatic orchestration.  
• Sub‑agents consume the same FeatureEnvelope; authenticity agent emits score+rationale; valuation agent consumes pricing fusion outputs to summarize.

# 5. Amazon Rekognition Integration (Pre‑Bedrock)

DevOps tasks:  
• Grant Lambda role permissions for Rekognition DetectText and DetectLabels (or Custom Labels if adopted later).  
• Ensure S3 bucket policies allow Rekognition read (via Lambda) for object bytes.  
• Configure quotas/limits; region alignment with other services; optional VPC endpoints if needed.

Runtime contract:  
• Input: S3 keys of uploaded images; Output: FeatureEnvelope JSON whose schema is versioned and stored in a shared contract.  
• FeatureEnvelope is stored transiently (Step Functions context) and minimally in DynamoDB for traceability.

# 6. Core AWS Resources

• API Gateway HTTP API: routes → Lambdas; JWT authorizer (Cognito). Stage logging/throttling.  
• Lambda functions: presign, cards CRUD, revalue entry, RekognitionExtract, PricingAgent, AuthenticityAgent, Aggregator.  
• DynamoDB: {stage}-CollectIQ table, on‑demand, PITR; GSIs for vault listing and analytics.  
• S3 uploads: private bucket w/ Block Public Access, SSE-S3, aws:SecureTransport, CORS for presigned PUT.  
• EventBridge: domain bus + rules + DLQ.  
• Step Functions: state machine JSON and IAM roles.  
• Amplify Hosting: Next.js SSR/ISR, custom domain app.collectiq.com, env var mapping to API.  
• Cognito: user pool + app client + Hosted UI; callback URLs restricted to Amplify domain & custom domain.  
• Secrets/SSM: external API keys & config.  
• X‑Ray & CloudWatch: tracing + metrics.

# 7. IAM & Security

• Per‑function IAM with minimum actions (S3, DDB, Rekognition, Bedrock, Step Functions, EventBridge).

• JWT authorizer for all routes except /healthz; CORS scoped to Amplify origin.

• S3 bucket policy enforces TLS and denies public access; presigned PUT only from backend.

• DDB policies limited to table/index ARNs; conditional writes for ownership enforcement handled in code.

• KMS encryption (S3/DDB/Secrets); secrets rotation schedule; audit logging enabled.

# 8. CI/CD

Backend pipeline: lint → typecheck → unit/integration → package Lambdas → upload artifacts → terraform plan/apply → smoke tests.  
Amplify pipeline: auto‑build + deploy on commits; PR previews enabled.  
Infra pipeline: terraform validate/plan → approval → apply; tflint/checkov; drift detection optional.  
Use Lambda aliases for canary/linear; rollback on CloudWatch alarm breach.

# 9. Observability & Runbooks

Dashboards: API 4xx/5xx & latency; Lambda p50/p95, errors, throttles; Step Functions executions; EventBridge DLQ; DDB throttles; Amplify deploy status; Bedrock/Reco invocation counts.

Alarms: Amplify build failures; API 5xx; Lambda error rate & duration; Step Functions failed exec; DLQ depth; DDB throttles; budget alarms.

Runbooks: alarm→service mapping; Rekognition/Bedrock quota increase steps; rollback; secrets rotation; rate‑limit playbook; Step Functions stuck execution handling.

# 10. Environments & Secrets

dev and prod with separate remote state and resource names. Secrets in Secrets Manager/SSM SecureString per env with strict IAM reads. Terraform outputs are consumed by Backend (API base URL, state machine ARN, Rekognition & Bedrock permissions context, etc.) and Frontend (Amplify URLs).

# 11. Interfaces & Outputs

Outputs to Backend: API base URL, Cognito IDs, DDB table & GSIs, S3 bucket, Step Functions ARN, EventBridge bus name, Rekognition permission context, Bedrock runtime access, JWKS URL.  
Outputs to Frontend: Amplify domain + custom domain, region, mapped env vars.