Proposed Architecture — PostNL Centralized Event Broker

Overview

The PostNL Event Broker is a serverless, event-driven integration platform enabling decoupled communication between PostNL applications. It provides: - Self-service onboarding for producers and consumers
- Guaranteed schema validation for event compatibility
- Multi-protocol interoperability (SQS, SNS, HTTPS, EventBridge)
- Observability and insight across all message flows
- Scalability and fault tolerance for millions of monthly events

2 High-Level Logical Architecture

Layer	AWS Services	Description
Authentication & Access	Amazon Cognito + CloudFront + API Gateway	Secured access to the self-service APIs
Self-Service APIs	AWS Lambda (broker_admin, consumer_admin, event_schema_validator)	Handles producer/ consumer registration and schema validation
Data Stores	DynamoDB	Schemas, producer catalog, subscriptions
Ingress Layer	SQS / API Gateway / EventBridge (ingress bus)	Accepts producer messages
Processing Layer	Lambda runtime_event_validator + EventBridge (core bus)	Validates & routes events
Egress Layer	SNS / SQS / HTTPS	Delivers validated events to consumers
Observability	CloudWatch + Grafana (optional)	Metrics & logs
Reliability	DLQs + retries + idempotency	Ensures message durability

3 Data Flow Summary

Step	Description	Responsible Component
1	Producer authenticates via Cognito and CloudFront	Cognito, API Gateway
2	Registers schema + ingress type	broker_admin, event_schema_validator
3	Consumer subscribes to event	consumer_admin
4	Producer sends event to SQS	Producer App → sqs_ingress_forwarder
5	Ingress bus receives event	EventBridge Ingress Bus
6	Runtime validation	runtime_event_validator, DynamoDB
7	Valid → Core bus → Consumer SNS	EventBridge, SNS
8	Invalid → DLQ	Runtime Validator, SQS DLQ
9	Insight metrics	insight_collector, CloudWatch

4 Reliability and Scalability

Feature	Mechanism
Schema validation	Dual-stage (creation + runtime)
Fault isolation	DLQs per Lambda stage
Retry policy	EventBridge 185 retries / 24 h
HA / DR	Multi-AZ + DynamoDB global tables (optional)
Throughput	EventBridge → 20 000 events/s target
Monitoring	CloudWatch logs + metrics + Grafana

5 Observability and Insight

- CloudWatch Logs and Metrics
- Grafana dashboards (via CloudWatch data source)
- DLQ depth monitoring
- Alarms on validation failures and latency

6 CI/CD and Testing

- Source: GitLab repo
- Build: Lint + Unit Test (Lambda pytest)
- Deploy: AWS CDK → CloudFormation

- Integration Tests: Schema and E2E validation
- Monitoring: Canary tests via CloudWatch Synthetics

7 Security

Concern	Control
AuthN/AuthZ	Cognito User Pools + IAM roles per Lambda
Encryption	KMS for SQS/SNS/DynamoDB
Secrets	AWS Secrets Manager
Compliance	CloudTrail and S3 log archiving

8 High Availability & Disaster Recovery

- Multi-AZ (default for serverless services)
- DynamoDB Global Tables for schemas replication
- EventBridge bus replication cross-region
- RTO ≈ 5 min, RPO ≈ 1 min

9 Alignment with the AWS Well-Architected Framework

Pillar	Applied Design Principles	How the Architecture Satisfies It
1. Operational Excellence	Infrastructure as Code (CDK), CI/CD, versioned schemas	Automated deployment, rollback, monitoring
2. Security	Cognito, IAM least privilege, KMS encryption, Secrets Manager	Secure data at rest/in transit, access control
3. Reliability	Multi-AZ, retries, DLQs, idempotency	Fault tolerance and predictable recovery
4. Performance Efficiency	Serverless, autoscaling	Elastic scaling to handle 20,000+ events/s
5. Cost Optimization	Pay-per-use serverless services	Costs scale with usage, minimal waste

Requirement Coverage

Requirement	Fulfillment
Self-service	Cognito + API Gateway + Lambdas
Reliability	JSON schema validation + DLQs
Compatibility	SQS/SNS/HTTPS/EventBridge
Insight	CloudWatch + Grafana
Scalability	Serverless auto-scaling
Security	IAM + KMS + Cognito
Monitoring	Logs, metrics, alarms
Automation	IaC + CI/CD
Demo	Python Lambdas + test payloads