CHURNGUARD™ REVOLUTIONIZING REVENUE OPERATIONS

A Technical Blueprint for Predictive Intelligence in Enterprise SaaS

Technical Design Document (TDD)

Project Overview

Project: ChurnGuard™: Predictive RevOps Intelligence Engine
Client: Enterprise SaaS (Tier 1)
Architect: Alex Rojas Segovia, CEO - Aineurolytics

Section 1: Executive Diagnosis

1.1 Context: Operational Friction

The Tier 1 Enterprise SaaS client is currently experiencing a quarterly customer churn rate of 8.5%. Retention efforts are reactive, initiated after a signal (e.g., failed renewal discussions, reduced usage). This reactive posture results in sub-optimal Sales and Customer Success resource allocation, where high-value customers exhibiting pre-churn indicators are treated identically to stable accounts. The estimated $ impact of this friction is $12.75 M in lost Annual Recurring Revenue (ARR) per quarter, requiring an 8.5% higher Customer Acquisition Cost (CAC) to maintain net revenue parity. This operational latency is a critical financial vulnerability.

1.2 The Solution: Scientific RevOps Architecture

ChurnGuard™ is a Scientific RevOps Intelligence Engine architected to provide proactive, weighted churn risk scoring and prescriptive actions. The solution employs a gradientboosted machine (XGBoost) core trained on behavioral, transactional, and engagement features to predict churn probability within a 90-day window. Orchestration via Airflow/Prefect ensures daily data ingestion and model inference, feeding high-fidelity risk scores **($\hat{y}_{risk} \in [0.0, 1.0]$)** directly into operational platforms (Salesforce, HubSpot) via a low-latency FastAPI microservice. This architecture transforms the retention strategy from reactive to predictive and resource-optimized.

Section 2: Strategic Impact & Projected Results

2.1 Strategic Context

Deployment of ChurnGuard™ shifts the Customer Success paradigm from universal coverage to risk-prioritized intervention. By surfacing a validated Churn Risk Score and key Feature Importances, the platform empowers RevOps teams to allocate high-touch resources exclusively to accounts with a **≥ 70 %** predicted churn risk, while automating low-touch interventions for moderate-risk accounts. This ensures maximum leverage from human capital, directly impacting the Net Revenue Retention (NRR) and reducing operational expenditure associated with broad-spectrum outreach.

2.2 KPI Table

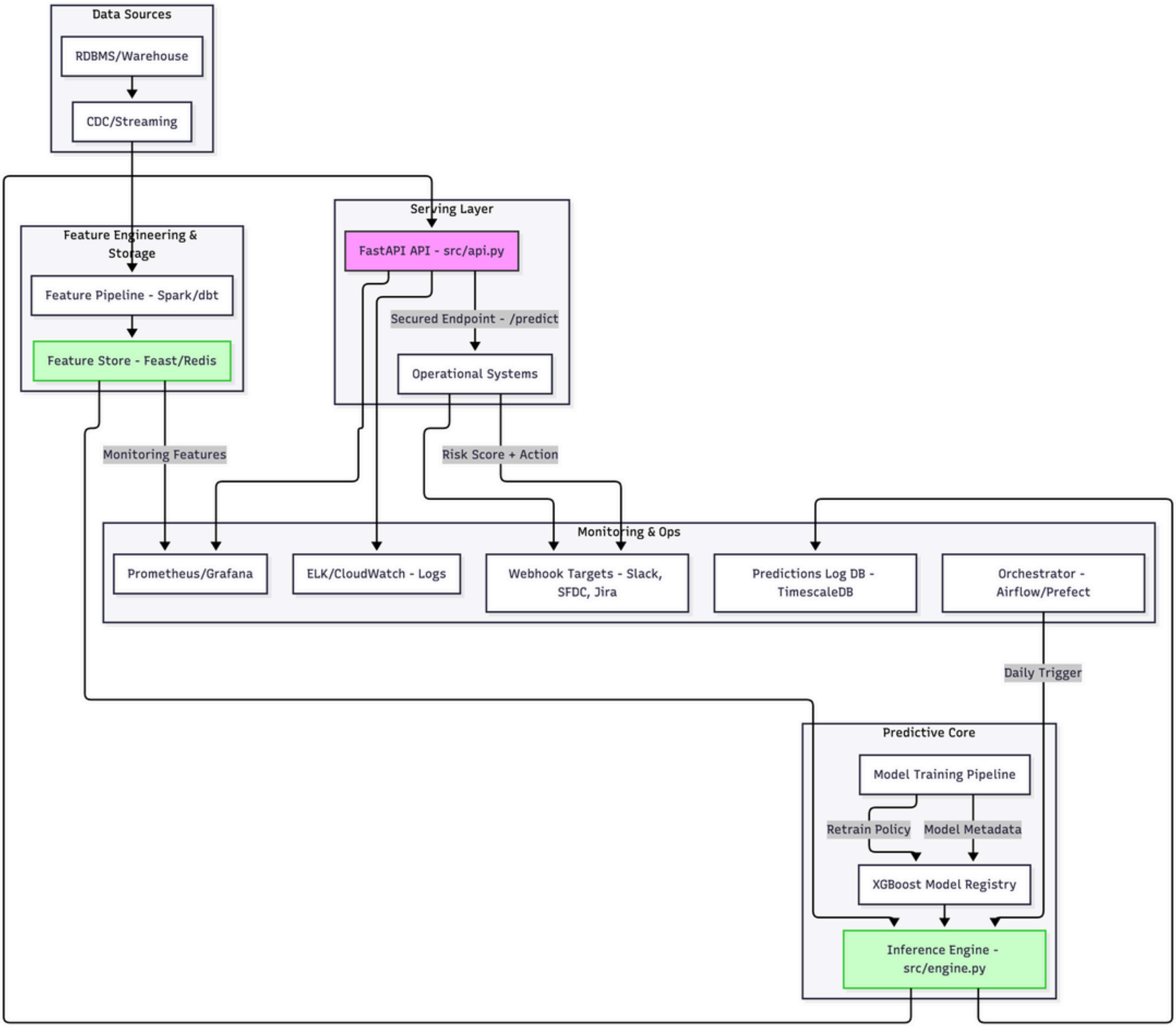
Metric Name	Baseline	Target	Projected Lift / ROI
Quarterly Customer Churn Rate	8.5%	4.0%	**53% Reduction**
ARR Protected (Qtrly)	$0	$5.4M	**$21.6M Annualized ARR Protection**
Customer Success Resource ROI	1.1x	1.8x	**64% Improvement in Efficiency**
Churn Prediction AUC Score	N/A	≥ 0.88	**High-Fidelity Predictive Power**
Time-to-Intervention (High-Risk)	14 days (Post-Signal)	24 hours (Pre-Signal)	**90%+ Reduction in Latency**
Customer Lifetime Value (CLV)	$150,000	$165,000	**10 % CLV Increase / ≈ $30M Impact**

SECTION 3: SYSTEM ARCHITECTURE

3.1 High-Level Design Overview

The architecture is designed for scalability and resilience. Ingestion is performed via CDC (Change Data Capture) from primary RDBMS (PostgreSQL/Snowflake) into a cloud-native Feature Store (Feast/DynamoDB). The Model Engine (containerized XGBoost) pulls features, performs inference, and persists the **($\hat{y}_{risk}$)** score to a dedicated Predictions Log (e.g., TimescaleDB). A low-latency FastAPI API acts as the serving layer, secured by OAuth2/JWT, offering the predict and metrics endpoints. Orchestration (Airflow/Prefect) manages the ETL, training, and deployment pipelines. All components emit structured logs to ELK/CloudWatch and metrics to Prometheus.

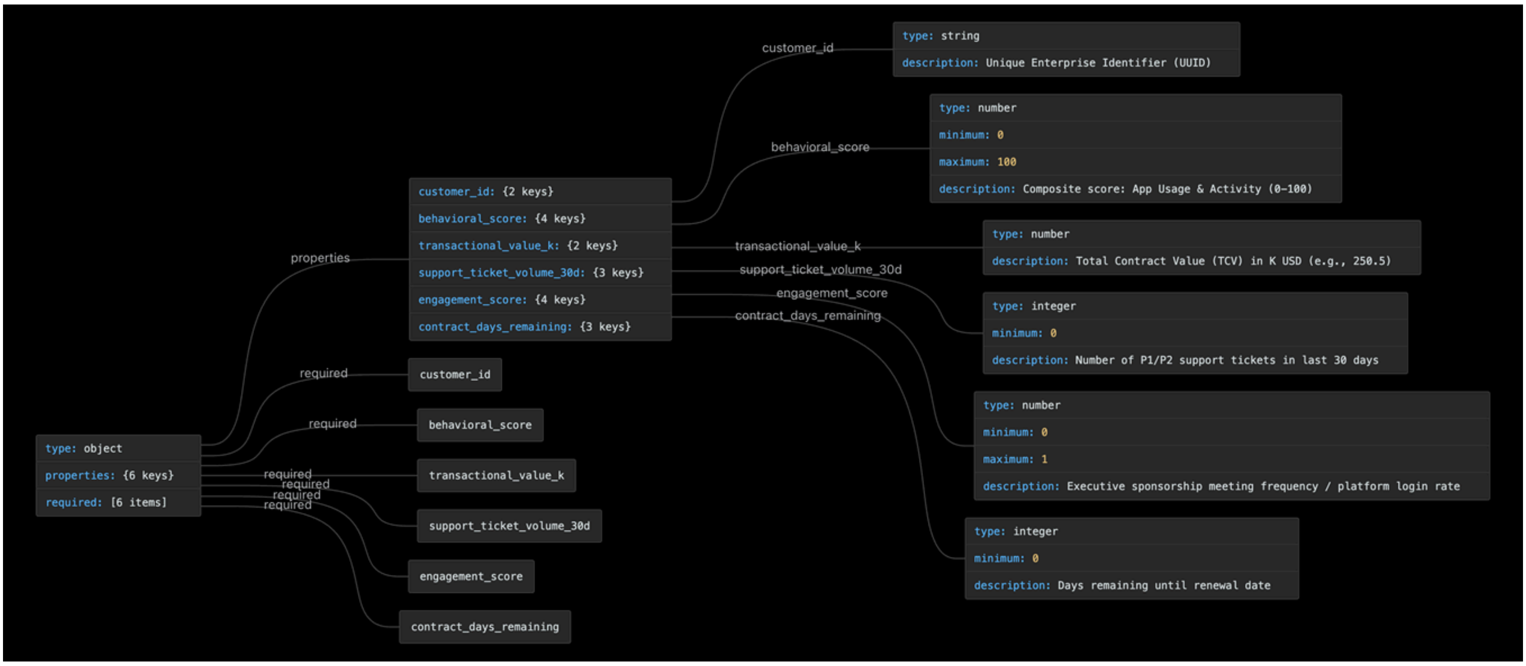
3.2 Diagram



3.3 Data Schema

The inference service accepts a payload for a single customer or a batch. The schema enforces strict typing for the 3 key feature categories.

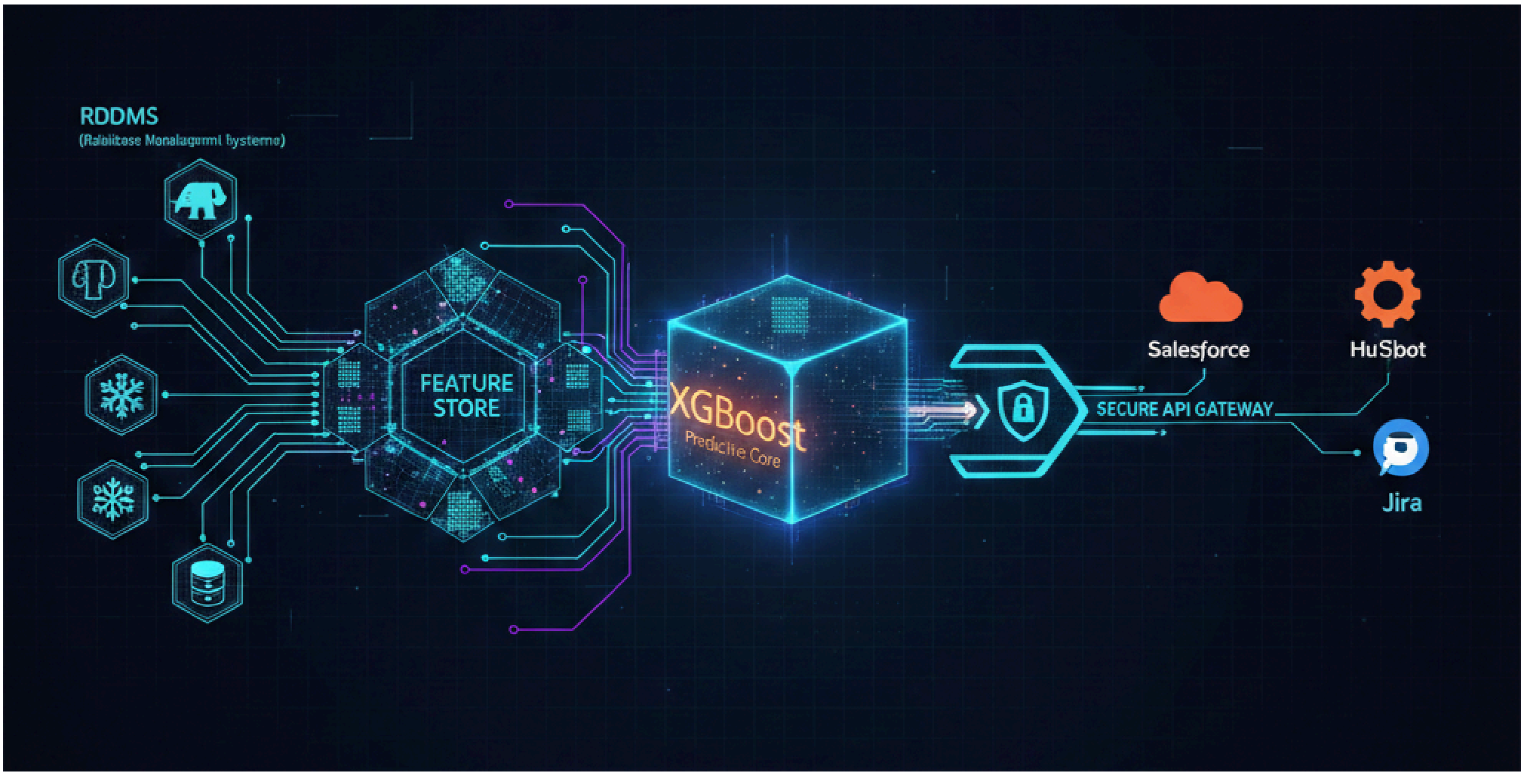
JSON Schema (InputFeatures)



Example JSON Payload

customer_id: cust-e7d8b9f0-c1a2-4d3e-b5f6-7a8b9c0d1e2f
behavioral_score: 45.8
transactional_value_k: 550
support_ticket_volume_30d: 7
engagement_score: 0.35
contract_days_remaining: 65

3.4 Visual Asset Placeholder



Section 4: Predictive Core (Algorithm)

The predictive core is implemented as a production-ready Python class, ChurnPredictor, using XGBClassifier (or a mock loader for the scaffold) to ensure model performance, reliability, and Aineurolytics standards adherence.

[Find the predictive core algorithm on GITHUB]

Section 5: Orchestration Layer

The orchestration layer is responsible for the end-to-end reliability of the predictive pipeline and the business-critical delivery of insights.

1. **Workflows and Triggers:**
   - **Training Pipeline:** Monthly **cron** trigger (or event-driven upon model drift detection) for full model retraining.
   - **Inference Pipeline: Daily cron** trigger (02:00 UTC) to ingest the previous day's data, run batch inference, and update the Predictions Log. Real-time inference is handled

by the low-latency API endpoint.

  - **Operational Sync:** Hourly micro-batch processing of the Predictions Log to send new/updated high-risk scores to downstream systems.

2. **Decision Gates:**
   - **Data Quality Gate:** Checks for **≥ 99.5 %** completeness and acceptable feature distribution (e.g., z-score outliers). Pipeline halts on failure, alerting the DataOps team.
   - **Model Performance Gate: AUC ≥ 0.88** post-retraining. If the new model fails, the pipeline rolls back to the previous, validated model version.

3. **Webhook Destinations:**
   - **Salesforce/HubSpot: Highest Priority**. Integration via direct API (or middleware) to update the **Churn_Risk_Score__c** field and trigger a Task creation for the assigned Account Executive/Customer Success Manager for accounts in the **HIGH** risk tier.
   - **Slack:** Low-priority notifications for pipeline status (success/failure) and **MEDIUM** risk account summary digest for team awareness.
   - **Jira: Automated P1 ticket generation** for critical system failures (e.g., API latency **≥ 500 ms** or Data Quality violation).

4. **Retry/Reconciliation Logic:**
   - All downstream API calls (e.g., Salesforce updates) implement **exponential backoff with Jitter** for transient errors (HTTP 429, 503). Max retries: 5.
   - **Deduplication:** The Predictions Log serves as the source of truth, ensuring idempotency and preventing duplicate alerts to CRMs based on the customer_id and prediction timestamp.

5. **Message Queue Considerations (Kafka/SQS):**
   - A **Kafka** topic is utilized for high-volume, real-time ingestion of granular behavioral data (if implemented in V2), decoupling the streaming ingestion from the batch processing.
   - **AWS SQS** can be used as a dead-letter queue (DLQ) for failed CRM webhook deliveries to allow for manual inspection and reconciliation by an MLOps engineer.

Section 6: MLOPS & Lifecycle

Robust MLOps practices are mandated to ensure the model maintains predictive validity and operational stability.

1. **Evaluation Metrics:**
    - **Primary Metric: Area Under the Receiver Operating Characteristic Curve (AUC)**. Target: ≥ 0.88. Provides overall discriminatory power.
    - **Secondary Metrics: Precision, Recall, and F1-Score** are monitored for the **HIGH Risk** class (**P ≥ 0.70**). High Recall is prioritized (i.e., minimize false negatives, ensuring we catch most churners), balanced with Precision to avoid alert fatigue.
    - **Calibration: Isotonic Regression** is used to ensure the predicted probability is a true reflection of the likelihood of churn, monitored by the **Reliability Diagram**.

2. **Drift Detection Approach:**
    - **Feature Drift:** Monitored daily using **Kolmogorov-Smirnov (KS) tests** or **Population Stability Index (PSI)** on key features (e.g., support_ticket_volume_30d). Alert threshold: PSI > 0.20 or KS P-value < 0.05.
    - **Concept Drift:** Monitored quarterly by re-evaluating the model's performance on recent, labeled data. Significant drop in AUC > 0.03 triggers an automatic retraining and MLOps review.

3. **Retraining Policy:**
    - **Scheduled Retraining:** Full training pipeline executed monthly.
    - **Event-Driven Retraining:** Triggered immediately upon confirmed **Feature or Concept Drift** exceeding thresholds or upon major shifts in the client's business logic/data structure.

4. **Model Registry (MLflow):**
    - **MLflow** is used to track all experimental runs, parameters, metrics, and to version control the serialized model and preprocessing artifacts (scaler, one-hot encoders). Only registered, staged models are eligible for deployment.

5. **Deployment Strategy:**
    - **Shadow Deployment:** The newly trained model is initially deployed in **Shadow Mode**, running parallel inference on production data. Its predictions are logged but not used for business action. Performance metrics (**AUC**, **Precision**) are compared against the current production model.
    - **Canary Promotion:** Once the Shadow Model demonstrates ≥ 99 % stability and statistically superior performance (**$AUC_{ne}w \geq AUC_{ol}d$**), it is promoted to a **Canary Deployment** (e.g., 10% of live traffic). Full promotion to 100% only occurs after 48 hours of stable Canary performance.

Section 7: Security, Compliance & Risk Modeling

Data security and regulatory adherence are non-negotiable for enterprise deployments.

1. **Authentication & Authorization:**
    - **API Access:** Secured via **OAuth 2.0 / JSON Web Tokens (JWT)**. All clients (CRMs, Orchestrator) must present a valid, non-expired JWT for access to /predict. Role-Based Access Control (RBAC) restricts access to sensitive endpoints (e.g., /admin).
    - **Service-to-Service:** Utilizes mutual TLS (**mTLS**) for communication between the API, Feature Store, and Orchestrator.

2. **Encryption:**
    - **Data in Transit:** All network communication is enforced via **TLS 1.2+**.
    - **Data at Rest:** All sensitive PII/Confidential Business Data (CBD) in the Feature Store and Predictions Log is encrypted using **AES-256** (e.g., AWS KMS or Azure Key Vault).

3. **Secrets Management:**
    - All API keys, database credentials, and webhook secrets are stored in a dedicated, audited secrets manager (**HashiCorp Vault or Cloud Secret Manager**) and injected into containers at runtime via environment variables, adhering to the principle of least privilege.

4. **SOC2/GDPR Considerations:**
    - **GDPR:** Prediction features are **pseudo-anonymized** (customer_id is an opaque UUID). The system avoids using strictly prohibited PII categories. Data retention policies are strictly enforced.
    - **SOC2:** Comprehensive audit logs (Section 9) track all data access, model training events, and deployment changes, providing the necessary evidence for SOC2 compliance.

5. **STRIDE Threat Matrix with Mitigation Actions:**

Threat Category	Example Threat	Mitigation Action
Spoofing	Unauthorized user gains API access.	Enforce OAuth2/JWT and strict RBAC. Use mTLS for internal services.
Tampering	Prediction score maliciously altered in DB.	Immutable Audit Logs (WORM storage) and DB-level access controls/encryption.
Repudiation	Engineer denies initiating a deployment change.	Mandatory Git/CI/CD history, full audit logging of deployment activity.
Information Disclosure	Sensitive features leaked via API.	Data encryption at rest/transit. Sanitization/filtering of API response payload.
Denial of Service (DoS)	Malicious request flood overwhelms the API.	Kubernetes HPA (Horizontal Pod Autoscaling), Rate Limiting, and WAF protection.
Elevation of Privilege	Low-privilege service account accesses secrets.	Least Privilege Principle, Secrets Management via Vault, continuous vulnerability scanning.

Section 8: DEVOPS, CI/CD & Deployment

The deployment strategy leverages modern, containerized, and declarative infrastructure for reliability and scale.

1. **Docker + Multi-Stage Builds:**
    - **Multi-stage Dockerfiles** minimize the final image size and reduce the attack surface (e.g., separate build stage for pip install and a slim runtime stage).

    - Base images are derived from minimal, security-hardened distributions (e.g., python:3.11-slim-buster).

2. **Kubernetes Deployment Pattern:**
    - **Platform:** Amazon EKS (Elastic Kubernetes Service) or GCP GKE.
    - **Pattern:** Microservice pattern with dedicated Deployment for the **FastAPI API** and a CronJob for the batch inference pipeline.
    - **Service Mesh:** Optional **Istio/Linkerd** for traffic shifting (Canary/Shadow) and mTLS.
3. **Helm:**
    - **Helm Charts** are used to define the application's Kubernetes resources (Deployment, Service, HPA, ConfigMap) in a reproducible, version-controlled manner.
4. **HPA Settings:**
    - **Horizontal Pod Autoscaler (HPA)** targets set to scale API pods based on resource utilization:
        - CPU Utilization: Target ≤ 65 %
        - Custom Metric: Target requests_per_second (Prometheus metric) ≥ 150.
    - Min Replicas: 3. Max Replicas: 15.
5. **GitHub Actions Pipeline:**

Stage	Trigger	Action
Lint & Test	push (all branches)	flake8, mypy, pytest (unit/integration tests).
Build & Push	push (main/release branch)	Docker multi-stage build, scan image (e.g., Clair), push to ECR/GCR.
Deploy (Non-Prod)	push (develop branch)	Helm deployment to Staging environment (runs smoke tests).
Deploy (Prod)	push (main/tag)	Helm deployment to Production (Shadow/Canary strategy).

1. **Rollback Strategy:**
   - **Kubernetes Rollback:** If a deployment fails health checks (Liveness/Readiness probes) or SLOs are violated, the CI/CD pipeline immediately initiates a helm rollback <release-name> <previous-revision>.
2. **SLO/SLA Definitions:**
   - **Latency (SLO):** 95 % of /predict requests must respond in ≤ 150 ms.
   - **Availability (SLA):** 99.95 % Annual Uptime (target: ≤ 4.38 hours downtime/year).
   - **Error Budget:** ≤ 0.05 % of all production requests can return a non-recoverable error (e.g., HTTP 5xx).

Section 9: Logging & Observability

A comprehensive observability stack ensures system health, traceability, and rapid incident response.

1. **Prometheus Metrics to Expose:**
   - churnguard_api_request_total: Counter, labeled by endpoint (/predict, /metrics) and HTTP status code.
   - churnguard_inference_latency_seconds: Histogram, tracks the duration of model inference.
   - churnguard_prediction_count: Counter, labeled by churn_risk_level (HIGH, MEDIUM, LOW).
   - churnguard_model_version: Gauge, tracks the active model version (e.g., v1.2.5).
2. **Grafana Dashboards to Create:**
   - **Executive Dashboard: ARR Protected (daily)**, **HIGH Risk Account Volume**, and API Latency P95/P99.
   - **MLOps Dashboard:** Feature/Concept Drift alerts (PSI/KS scores), Model AUC/F1, and Retraining Pipeline status.
   - **DevOps Dashboard:** Kubernetes pod/node resource usage, HTTP error rates, and HPA activity.
3. **Trace IDs in Logs:**

- Every incoming request to the FastAPI API is assigned a unique **Trace ID** (UUID). This ID is propagated and prepended to every subsequent log entry across the entire request path (API -> Engine -> DB).

4. **ELK/CloudWatch Usage:**
   - **Structured Logging (JSON):** All application and infrastructure logs are emitted in a structured JSON format, ingested into **CloudWatch Logs / ElasticSearch**.
   - This facilitates efficient querying/filtering by log_level, trace_id, and customer_id.

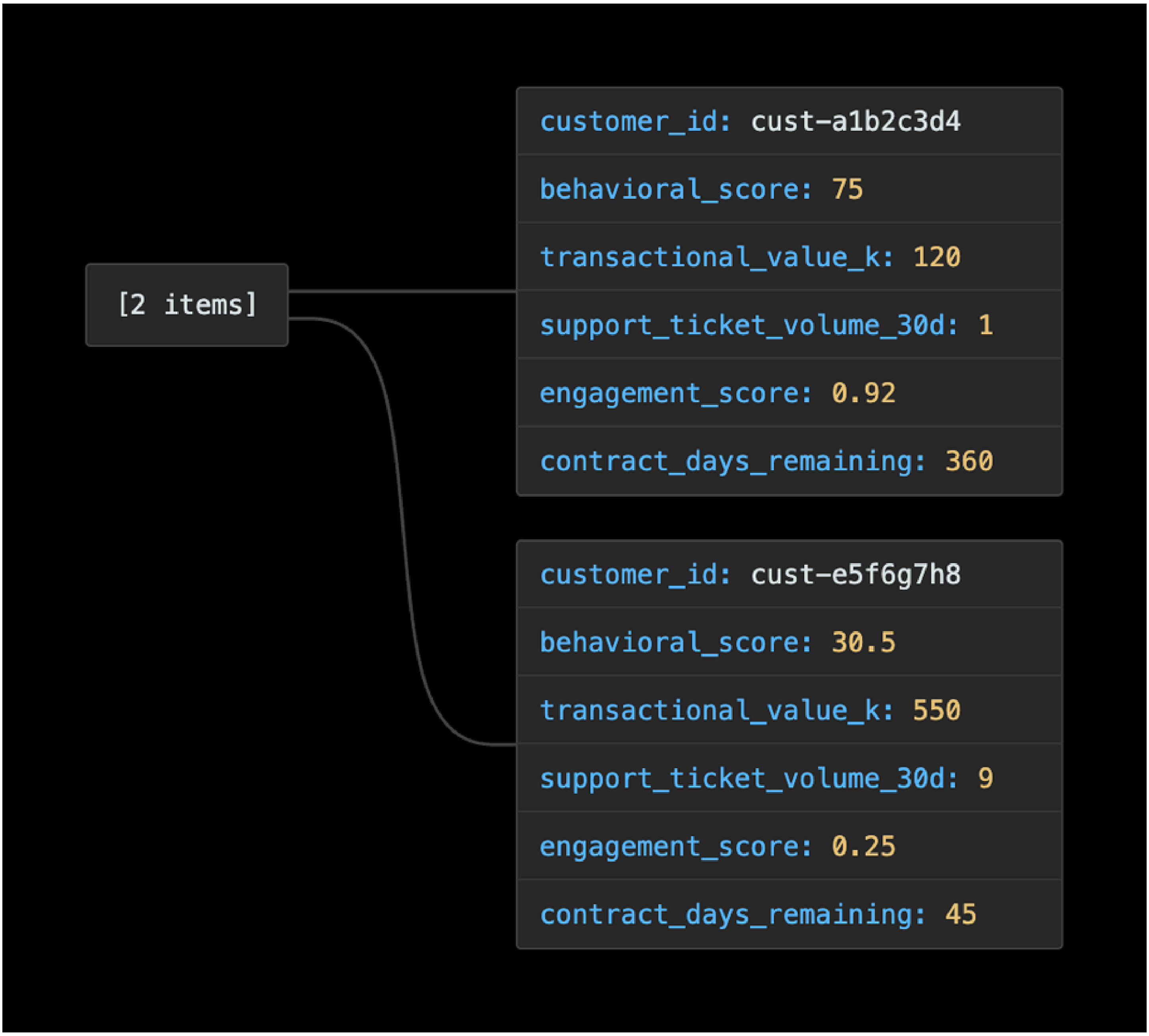
5. **Alert Rules and Playbooks:**
   - **Critical Alert (P1):** API Error Rate **> 0.1 %** for 5 minutes. **Action:** PagerDuty/On-Call Escalation, automatic rollback.
   - **High Alert (P2):** Model AUC drops by > 0.03 during Shadow Deployment. **Action:** JIRA ticket to MLOps team, Slack alert.
   - **Warning Alert (P3):** Feature Drift (PSI > 0.15). **Action:** Slack warning to Data Ops, scheduled review.
   - **Playbooks:** Detailed, version-controlled runbooks outlining diagnostic steps, mitigation actions, and communication protocols for each alert type.

APPENDIX

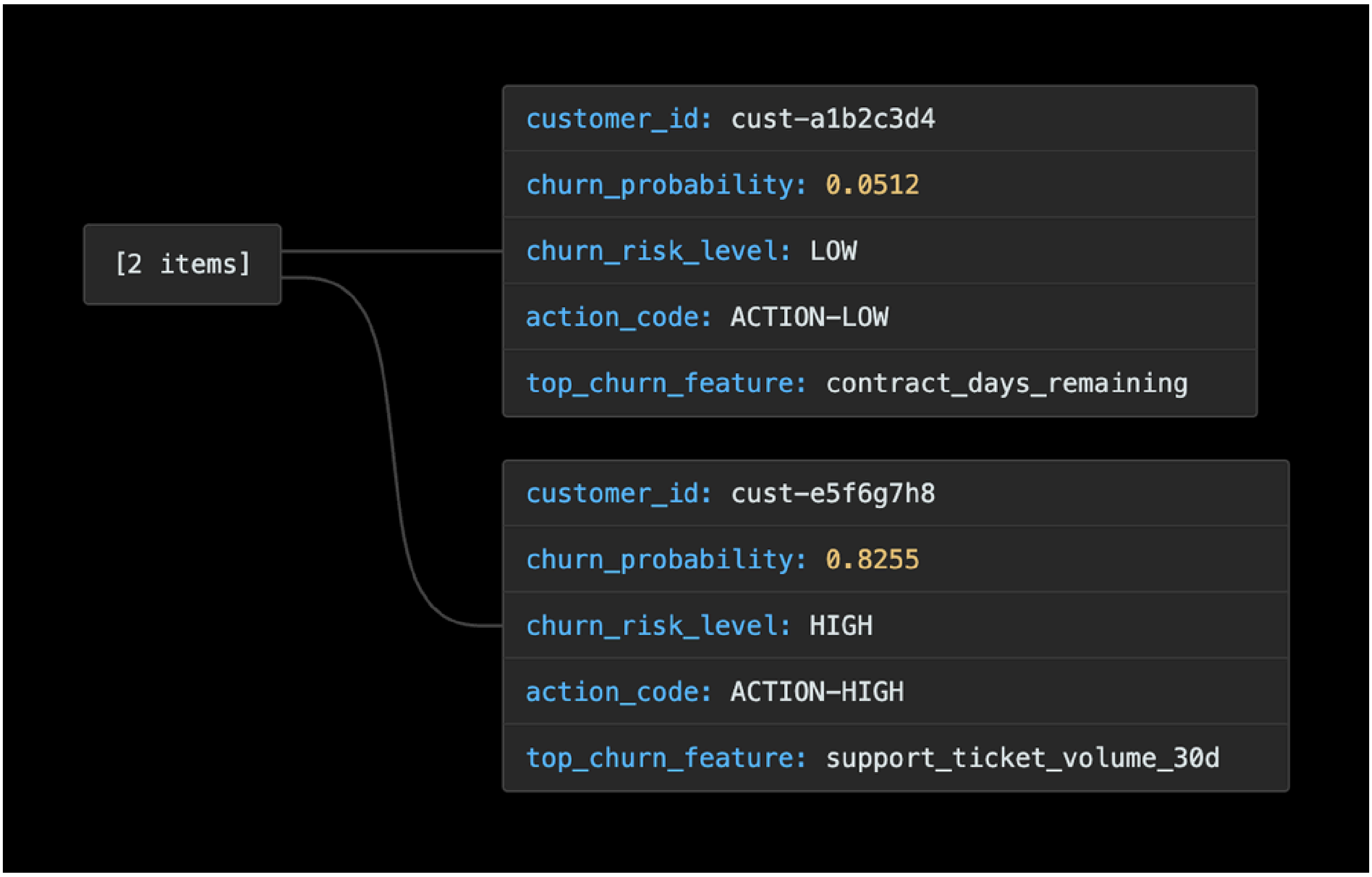
API Contract Example: POST /predict

- **Endpoint:** /api/v1/predict
- **Method:** POST
- **Content-Type:** application/json
- **Authorization:** Bearer <JWT>

Sample JSON Request



Sample JSON Response (HTTP 200 OK)



Simple SQL DDL for Predictions Log Table

SQL

```
CREATE TABLE predictions_log (
  prediction_id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  customer_id VARCHAR(128) NOT NULL,
  prediction_timestamp TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
  churn_probability REAL NOT NULL,
  churn_risk_level VARCHAR(10) NOT NULL,
  model_version VARCHAR(32) NOT NULL,
  inference_features JSONB NOT NULL,
  webhook_status VARCHAR(50) DEFAULT 'PENDING'
);

CREATE INDEX idx_customer_time ON predictions_log (customer_id, prediction_timestamp DESC);
```

RACI Matrix for Deployment Responsibilities

Activity	**DevOps Engineer**	**MLOps Engineer**	**Enterprise Architect**	**CEO (Alex Rojas)**
Infrastructure Provisioning (EKS/Helm)	Responsible/Accountable	Consulted	Informed	Informed
Model Deployment & Validation	Consulted	Responsible/Accountable	Informed	Informed
CI/CD Pipeline Maintenance	Responsible	Responsible	Informed	Informed
Security & Compliance Review	Informed	Informed	Accountable	Consulted
Strategic Business Alignment	Informed	Informed	Responsible	Accountable