TECHNICAL ARCHITECTURE DOCUMENT

RIGACAP

System Architecture & Technical Audit

Comprehensive technical documentation covering infrastructure, database schema, API surface, security architecture, deployment pipeline, and operational characteristics.

Confidential — February 2026

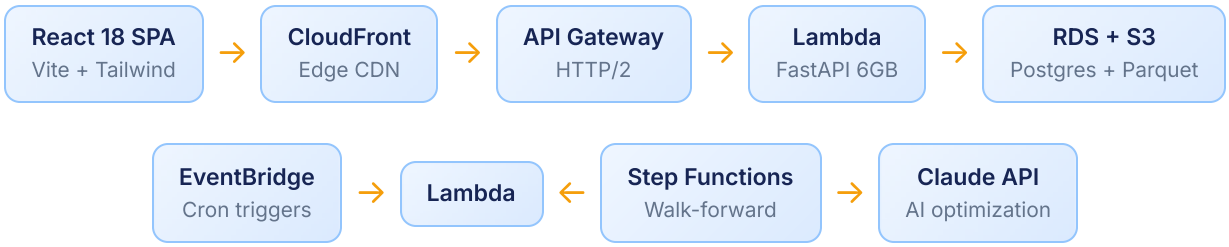
v2.5.0 · 95 endpoints · 18 tables · 15K+ LOC

Table of Contents

95	18	22+	15K+
API ENDPOINTS	DB TABLES	BACKEND SERVICES	LINES OF CODE

1. System Overview

RigaCap is a serverless, event-driven trading platform built on AWS. The system ingests market data from 6,500+ US equities, generates AI-powered trading signals, and delivers them to subscribers via a React dashboard and automated email pipeline. The entire backend runs on a single AWS Lambda function using FastAPI + Mangum for ASGI compatibility.



2. AWS Infrastructure

All infrastructure is managed via Terraform (`infrastructure/terraform/main.tf`). Resources are deployed in `us-east-1`.

Compute

RESOURCE	TYPE	CONFIGURATION
Lambda Function	`rigacap-prod-api`	Container image (ECR), 6GB RAM, 900s timeout, VPC-attached
API Gateway	HTTP API (v2)	Custom domain: `api.rigacap.com`, CORS configured, HTTP/2
ECR Repository	`rigacap-prod-api`	Mutable tags, scan-on-push, keep last 5 images
Step Functions	`rigacap-prod-walk-forward`	Orchestrates Lambda for multi-period walk-forward simulations

Storage

BUCKET	ACCESS	PURPOSE
`rigacap-prod-frontend-*`	Public (CloudFront origin)	React SPA static files (~1-2MB)

BUCKET	ACCESS	PURPOSE
`rigacap-prod-price-data-*`	Private (Lambda IAM only)	Parquet price cache, chart cards, social images (50-100MB)

Database

RESOURCE	CONFIGURATION
RDS PostgreSQL 15	`db.t3.micro`, 20GB gp2, deletion protection enabled, asyncpg driver
Connection Pool	Pool size: 5, max overflow: 10, connect timeout: 3s, query timeout: 5s

Scheduled Events (EventBridge)

RULE	SCHEDULE	TARGET
Market Scanner	`cron(0 21 ? * MON-FRI *)` (4 PM ET)	Lambda → `/api/signals/scan`
Lambda Warmer	`rate(5 minutes)`	Lambda → `/health`

3. CDN, DNS & SSL

CloudFront Distribution

SETTING	VALUE
Domains	`rigacap.com`, `www.rigacap.com`
Origin	S3 website endpoint (HTTP-only)
Protocol	Redirect HTTP → HTTPS
Price Class	PriceClass_100 (US, Canada, Europe)
Default TTL	1 hour (max 1 day)
SPA Routing	Custom error: 404 → `/index.html` (200)
SSL Certificate	ACM: `rigacap.com` + `*.rigacap.com` (DNS validated)

Route53 DNS

RECORD	TYPE	TARGET
`rigacap.com`	A (alias)	CloudFront distribution
`www.rigacap.com`	A (alias)	CloudFront distribution
`api.rigacap.com`	A (alias)	API Gateway custom domain

4. Database Schema

PostgreSQL 15 with async driver (`asyncpg`) via SQLAlchemy 2.0 async sessions. All tables use auto-increment integer primary keys except `users` and `subscriptions` (UUID).

Core Data Tables

stock_data

Daily OHLCV price data with computed indicators. One row per symbol per trading day.

`id` INTEGER PK

`symbol` VARCHAR(10) IDX

`date` DATETIME IDX

`open, high, low, close` FLOAT

`volume` FLOAT

`dwap, ma_50, ma_200` FLOAT

signals

Trading signals generated by the scanner. Status lifecycle: active → executed/expired.

`id` INTEGER PK

`symbol` VARCHAR(10) IDX

`signal_type` VARCHAR(10) BUY/SELL

`price, dwap, pct_above_dwap` FLOAT

`volume, volume_ratio` FLOAT

`stop_loss, profit_target` FLOAT

`is_strong` BOOLEAN

`status` VARCHAR(20)

`created_at, expires_at` DATETIME

positions

Open/closed portfolio positions with trailing stop tracking.

`id` INTEGER PK

`user_id` UUID FK→users

`symbol` VARCHAR(10) IDX

`entry_date, entry_price` DATETIME, FLOAT

`shares, stop_loss` FLOAT

`profit_target, highest_price` FLOAT

`signal_id` INTEGER FK→signals

`status` VARCHAR(20) open/closed

trades

Completed trade records with P&L tracking.

`id` INTEGER PK

`user_id` UUID FK→users

`position_id` INTEGER FK→positions

`symbol` VARCHAR(10) IDX

`entry_date, exit_date` DATETIME

`entry_price, exit_price` FLOAT

`pnl, pnl_pct` FLOAT

`exit_reason` VARCHAR(50)

Authentication & Subscriptions

users

User accounts with multi-provider OAuth support. UUID primary keys.

- `id` UUID PK
- `email` VARCHAR(255) UNIQUE IDX
- `password_hash` VARCHAR(255) NULLABLE
- `name` VARCHAR(255)
- `role` VARCHAR(20) admin/user
- `google_id, apple_id` VARCHAR(255) UNIQUE
- `stripe_customer_id` VARCHAR(255) UNIQUE
- `is_active` BOOLEAN
- `created_at, last_login` DATETIME

subscriptions

Stripe-managed subscription state. Status: trial/active/canceled/expired/past_due.

- `id` UUID PK
- `user_id` UUID FK→users UNIQUE
- `status` VARCHAR(20)
- `stripe_subscription_id` VARCHAR(255)
- `stripe_price_id` VARCHAR(255)
- `trial_start, trial_end` DATETIME
- `current_period_start/end` DATETIME
- `cancel_at_period_end` BOOLEAN

Strategy Management

strategy_definitions

Trading strategy configurations. Types: dwap, momentum, ensemble.

- `id` INTEGER PK
- `name` VARCHAR(100) UNIQUE
- `strategy_type` VARCHAR(50)
- `parameters` TEXT (JSON)
- `is_active` BOOLEAN IDX
- `source` VARCHAR(50) manual/ai
- `is_custom` BOOLEAN

strategy_evaluations

Performance evaluations for strategies

`strategy_id` FK→strategy_definitions

`total_return_pct, sharpe_ratio` FLOAT

`max_drawdown_pct, win_rate` FLOAT

`recommendation_score` FLOAT 0-100

strategy_switch_history

Audit trail for strategy switches

`from/to_strategy_id` FK→strategy_definitions

`trigger` manual/auto_scheduled

`score_before, score_after` FLOAT

`reason` TEXT

Simulation & Backtesting

walk_forward_simulations

Walk-forward simulation runs with aggregate results. Trades stored as JSON to avoid 256KB Step Functions state limit.

`id` INTEGER PK

`reoptimization_frequency` VARCHAR(20)

`max_drawdown_pct` FLOAT

`equity_curve_json` TEXT (JSON)

`is_daily_cache` BOOLEAN IDX

`start_date, end_date` DATETIME

`total_return_pct, sharpe_ratio` FLOAT

`trades_json` TEXT (JSON array)

`status` pending/completed/failed

social_posts

AI-generated social content with approval workflow. Status: draft → approved → scheduled → posted/canceled.

`id` INTEGER PK

`platform` twitter/instagram

`text_content, hashtags` TEXT

`scheduled_for, posted_at` DATETIME

`ai_model, ai_prompt_hash` VARCHAR

`source_trade_json` TEXT

`post_type` trade_result/missed_opp/recap

`status` VARCHAR(20)

`image_s3_key` VARCHAR(500)

`ai_generated` BOOLEAN

`news_context_json` TEXT

Model Portfolio & Intelligence Tables

model_positions

Model portfolio positions (live, walkforward, ghost variants). Tracks entry/exit with signal replay data.

- `id` INTEGER PK
- `portfolio_type` VARCHAR(20) IDX
- `symbol` VARCHAR(10) IDX
- `entry_date, entry_price` DATETIME, FLOAT
- `shares, cost_basis` FLOAT
- `highest_price, exit_price` FLOAT
- `exit_reason, status` VARCHAR
- `signal_data_json` TEXT (JSON)
- `autopsy_json` TEXT (JSON) — AI trade post-mortem

regime_forecast_snapshots

Daily regime forecast persistence for historical accuracy tracking and heatmap visualization.

- `id` INTEGER PK
- `snapshot_date` DATETIME UNIQUE IDX
- `current_regime` VARCHAR(30)
- `probabilities_json` TEXT (JSON) — 7 regime probabilities
- `outlook` stable/deteriorating/improving
- `recommended_action` VARCHAR(30)
- `spy_close, vix_close` FLOAT

5. Data Pipeline & Caching

yfinance Ingestion

Daily download of 252-day rolling OHLCV for 100+ symbols. Split/dividend adjusted. Free API, no key required.

S3 Parquet Cache

Price data persisted to S3 as parquet files. Lambda loads on startup via `/api/warmup` endpoint. 7-day TTL.

In-Memory Cache

ScannerService maintains `data_cache: Dict[str, DataFrame]` in Lambda memory. 600-800MB typical footprint.

CDN Pre-computation

Dashboard JSON exported to S3 daily. CloudFront serves with 1-hour TTL for zero-latency dashboard loads.

6. Backend Services

22+ singleton service classes comprising 15,000+ lines of Python. All services are instantiated at module level and imported globally. The Lambda function uses Mangum for ASGI-to-Lambda translation.

SERVICE	LOC	RESPONSIBILITY
ScannerService	940	Market scan, signal generation, data cache management
BacktesterService	1,255	Historical backtesting, portfolio simulation, performance metrics
WalkForwardService	1,907	Step Functions orchestration, period-by-period evaluation, AI optimization
EmailService	2,358	Subscriber emails: welcome, daily digest, sell alerts, password reset
SchedulerService	1,454	APScheduler background jobs, nightly digest, social post scheduling
ModelPortfolioService	1,200	Dual live/WF portfolio tracking, ghost portfolios, what-if calculator
MarketRegimeService	882	7-regime market classification using SPY indicators

SERVICE	LOC	RESPONSIBILITY
DataExportService	827	S3 data export/import, CDN pre-computation, parquet persistence
SocialContentService	759	Template-based post generation (fallback for AI service)
AutoSwitchService	683	Automated strategy switching with scoring and cooldown
StockUniverseService	461	Stock universe management, sector classification, ETF filtering
StrategyAnalyzerService	437	Multi-strategy comparison and performance attribution
StrategyGeneratorService	434	Optuna-based parameter optimization, AI strategy generation
ChartCardGenerator	346	1080×1080 matplotlib chart cards with brand styling
PostSchedulerService	292	Post scheduling, notification pipeline, cancel mechanism
AIContentService	287	Claude API integration for AI-powered content generation
SocialPostingService	285	Twitter API v2 + Instagram Graph API publishing
TradeAutopsyService	175	Claude-powered forensic post-mortem analysis for closed trades
RegimeForecastService	185	Daily regime forecast storage, accuracy tracking, heatmap data

Lambda Cold Start Mitigation: Database connections are initialized lazily on first request (Lambda init has a 10-second timeout). An EventBridge rule pings `/health` every 5 minutes to keep the Lambda warm. Post-deploy, `/api/warmup` preloads S3 data into memory.

7. API Surface (95 Endpoints)

Authentication (`/auth`) — 9 endpoints

METHOD	PATH	DESCRIPTION
`POST`	/auth/register	Email/password signup with Turnstile verification
`POST`	/auth/login	Email/password login, returns JWT access + refresh tokens
`POST`	/auth/refresh	Refresh access token using refresh token
`GET`	/auth/me	Current user profile
`POST`	/auth/google	Google OAuth signup/login
`POST`	/auth/apple	Apple Sign In signup/login
`POST`	/auth/forgot-password	Send password reset email
`POST`	/auth/reset-password	Reset password with secure token
`POST`	/auth/logout	Invalidate current tokens

Signals (`/signals`) — 14 endpoints

METHOD	PATH	AUTH
`GET`	/signals/scan	Subscriber (runs full market scan)
`GET`	/signals/latest	Subscriber
`GET`	/signals/dashboard	Subscriber (comprehensive data)
`GET`	/signals/momentum-rankings	Subscriber
`GET`	/signals/missed	Subscriber (missed opportunities)
`GET`	/signals/double-signals	Subscriber (multi-confirmation)
`GET`	/signals/watchlist	Subscriber
`GET`	/signals/approaching-trigger	Subscriber
`GET`	/signals/symbol/{symbol}	Subscriber

METHOD	PATH	AUTH
GET	/signals/info/{symbol}	Subscriber
GET	/signals/cdn-url	Subscriber
GET	/signals/memory-scan	Subscriber
POST	/signals/backfill	Admin
GET	/signals/simulate-intraday-crossover	Admin

Billing (`/billing`) — 5 endpoints

METHOD	PATH	DESCRIPTION
POST	/billing/create-checkout	Create Stripe Checkout session (7-day trial, CC required)
POST	/billing/portal	Create Stripe Customer Portal session
GET	/billing/subscription	Get current subscription status
POST	/billing/sync	Sync subscription state from Stripe
POST	/billing/webhook	Stripe webhook receiver (signature verified)

Social (`/social`) — 16 endpoints

METHOD	PATH	DESCRIPTION
GET	/social/posts	List posts (filter by status, type, platform)
POST	/social/posts/compose	Create new draft post
POST	/social/posts/{id}/approve	Approve for publishing
POST	/social/posts/{id}/reject	Reject with reason
POST	/social/posts/{id}/publish	Publish immediately
POST	/social/posts/{id}/schedule	Schedule for future date
POST	/social/posts/{id}/cancel	Cancel scheduled post
POST	/social/posts/{id}/regenerate-ai	Re-generate via Claude API
POST	/social/posts/{id}/edit	Edit draft content
DELETE	/social/posts/{id}	Delete draft

METHOD	PATH	DESCRIPTION
`GET`	/social/posts/{id}/cancel-email	One-click cancel via JWT link
`POST`	/social/generate-chart/{id}	Generate chart card image
`GET`	/social/stats	Publishing statistics

Admin (`/admin`) — 50+ endpoints

All admin endpoints require `role=admin` AND email in `ADMIN_EMAILS` allowlist. Categories include: strategy management (20+), user management (6), market regime analysis (4), walk-forward simulation (8), backtesting (2), data debugging (2), model portfolio (10), trade autopsies (2), ghost comparison (1), regime forecast (3), what-if calculator (1).

8. Frontend Architecture

React 18	Vite 5	Tailwind 3
UI FRAMEWORK	BUILD TOOL	CSS FRAMEWORK

Component Architecture

COMPONENT	SIZE	RESPONSIBILITY
App.jsx	171 KB	Main dashboard: portfolio, signals, charts, auth, time-travel (admin)
WalkForwardSimulator.jsx	67 KB	Configure/run simulations, equity curves, trade analysis
SocialTab.jsx	55 KB	Post queue, AI generation, scheduling, chart cards, publishing
FlexibleBacktest.jsx	39 KB	Backtest config, strategy comparison, exit testing
AdminDashboard.jsx	42 KB	Strategy/user management, model portfolio, ghost comparison, regime forecast, what-if calculator, trade autopsies
LandingPage.jsx	30 KB	Marketing: hero, features, pricing, FAQ, social proof
StrategyEditor.jsx	16 KB	Custom strategy creation and editing
AutoSwitchConfig.jsx	15 KB	Auto-switch rule configuration
LoginModal.jsx	14 KB	Multi-provider auth: email, Google, Apple, Turnstile
StrategyGenerator.jsx	12 KB	AI strategy generation interface
LegalPages.jsx	11 KB	Terms, Privacy Policy, Financial Disclaimer
PasswordReset.jsx	9 KB	Forgot/reset password flow
SubscriptionBanner.jsx	9 KB	Trial countdown, upgrade CTA

Key Frontend Libraries

- **Recharts** — Dashboard charts, equity curves
- **Axios** — HTTP client with JWT interceptor
- **React Router v6** — SPA routing
- **Lucide React** — SVG icon library
- **react-hot-toast** — Notification system
- **date-fns** — Date formatting/manipulation
- **clsx** — Conditional classname utility
- **Puppeteer** — Headless browser (chart rendering)

9. Security Architecture

Authentication Flow

LAYER	IMPLEMENTATION
Password Hashing	bcrypt via passlib (auto-configured cost factor)
JWT Tokens	HS256, 15-min access / 7-day refresh, python-jose library
OAuth Validation	Google: ID token sig via public keys. Apple: JWKS with 1hr cache
Bot Protection	Cloudflare Turnstile on registration endpoint
CORS	Whitelist: rigacap.com, www.rigacap.com, localhost dev ports
Admin Guard	Role check + ADMIN_EMAILS allowlist (double enforcement)
Subscription Gate	`require_valid_subscription` dependency on signal endpoints
Payment Security	Stripe Checkout (PCI DSS), webhook signature verification
S3 Access	Price data bucket: all public access blocked, IAM-only
HTTPS	ACM certificates, CloudFront redirect HTTP→HTTPS
Email Cancel Links	JWT-signed one-click cancel URLs for social post management

10. CI/CD Pipeline

Fully automated deployment via GitHub Actions on push to `main`. Zero-downtime updates via Lambda container image swaps.

```
# Trigger: push to main OR pull request to main

Stage 1: Test
  Backend: Python 3.11 → pip install → pytest -v
  Frontend: Node 18 → npm ci → npm run lint → npm test

Stage 2: Deploy (only on main push)
  1. npm run build → S3 sync → CloudFront invalidation
  2. Pre-deploy: export price data to S3
  3. Docker build (linux/amd64, --provenance=false) → ECR push
  4. Lambda update-function-code → new image
  5. Post-deploy: /api/warmup (load S3 data into memory)
```

> **Critical Lambda Build Requirement:** Docker BuildKit creates multi-platform manifests with attestations that Lambda rejects. All builds use `--provenance=false --sbom=false` flags for compatibility.

11. Monitoring & Performance

Current Monitoring

COMPONENT	METHOD	RETENTION
Lambda Logs	CloudWatch (Python logging)	30 days
API Gateway	CloudWatch metrics (4xx, 5xx)	14 days
Step Functions	CloudWatch log group (ERROR level)	30 days
Database	RDS console metrics	14 days

Performance Characteristics

OPERATION	LATENCY	RESOURCE
Dashboard load	<2 seconds	Pre-computed JSON from CDN
Signal scan (100+ symbols)	30–45 seconds	yfinance download + calculation
Single backtest (90d, 100 symbols)	20–40 seconds	Lambda 6GB memory
Walk-forward (5yr, 26 periods)	10–30 minutes	Step Functions → Lambda
AI content generation	2–5 seconds	Claude API (Sonnet 4.5)
Chart card generation	1–3 seconds	matplotlib (Lambda, 6GB)

Resource Utilization

6 GB	~800 MB	20 GB	~100 MB
LAMBDA MEMORY	TYPICAL USAGE	RDS STORAGE	S3 PRICE DATA

12. Dependencies

Backend (Python 3.11)

- **FastAPI** 0.109.0 + Uvicorn 0.27.0
- **SQLAlchemy** 2.0.25 + asyncpg 0.29.0
- **Pydantic** 2.5.0
- **pandas** 2.1.4 + numpy 1.26.3
- **yfinance** ≥1.1.0
- **Optuna** ≥3.5.0
- **matplotlib** ≥3.8.0
- **boto3** 1.34.0 + Mangum 0.17.0
- **Stripe** 7.10.0
- **python-jose** 3.3.0 + passlib 1.7.4
- **APScheduler** 3.10.4
- **httpx** 0.26.0 (Claude API client)

Frontend (Node 18)

- **React** 18.2.0
- **Vite** 5.0.0
- **TailwindCSS** 3.3.5
- **React Router** 6.20.0
- **Recharts** 2.10.0
- **Axios** 1.6.0
- **Lucide React** 0.292.0
- **Puppeteer** 24.37.3
- **ESLint** 8.53.0
- **Vitest** 0.34.6

Technical Architecture Document — v2.4.0 — February 2026

This document is confidential and intended solely for authorized technical reviewers. Infrastructure details should not be shared outside the organization.