AIAlgoTradeHits.com Agentic AI Architecture

Master Query & Configuration Guide

Version: 6.0
Last Updated: January 2026
Classification: Enterprise Architecture Specification
Platform: Google Cloud Platform (GCP)
AI Core: Vertex AI + Gemini 2.5 Pro + XGBoost Ensemble

TABLE OF CONTENTS

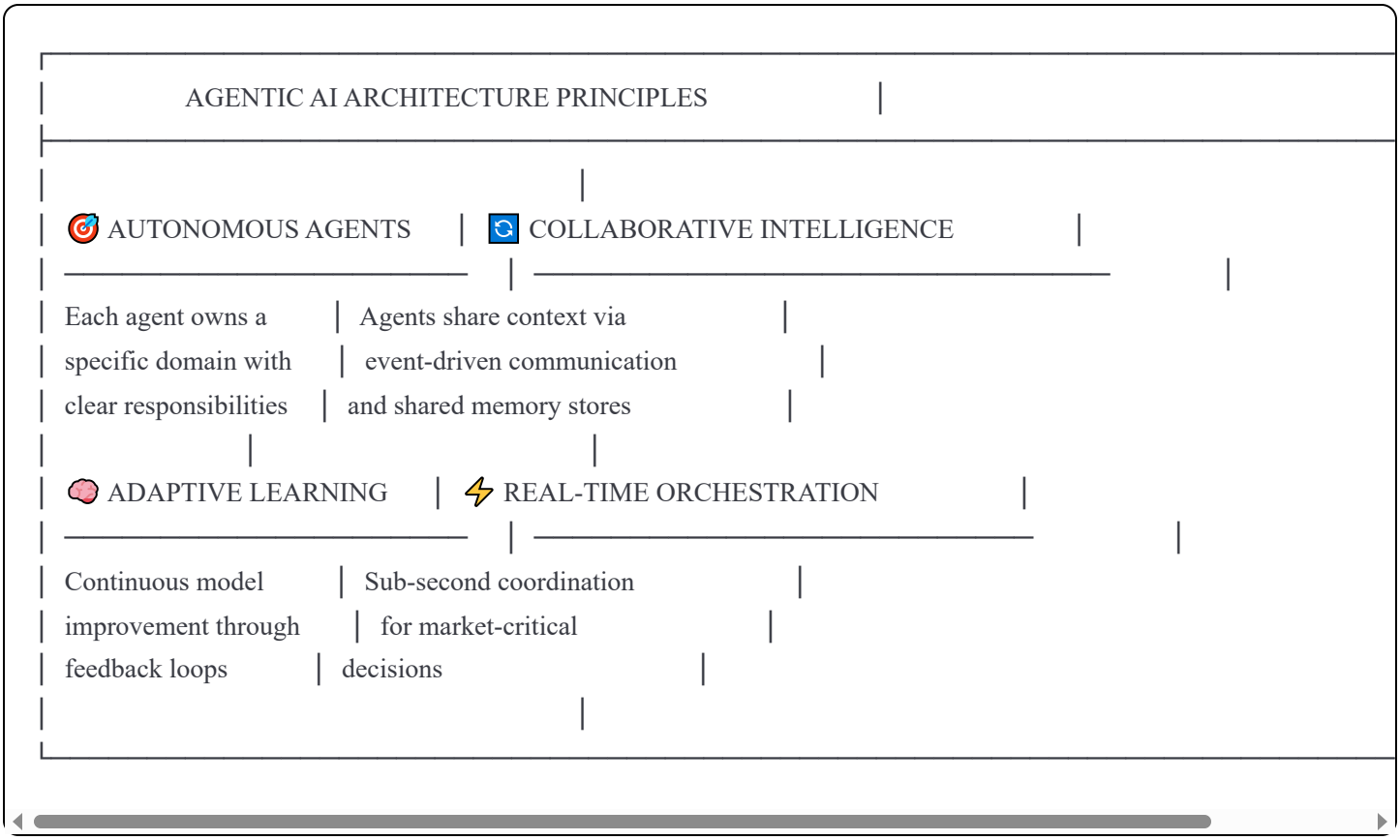
---

1. EXECUTIVE SUMMARY

1.1 Platform Vision

AIAlgoTradeHits.com is a comprehensive AI-driven trading analytics ecosystem built on **Agentic AI Architecture** - a paradigm where specialized autonomous agents collaborate to deliver institutional-grade trading intelligence across multiple asset classes.

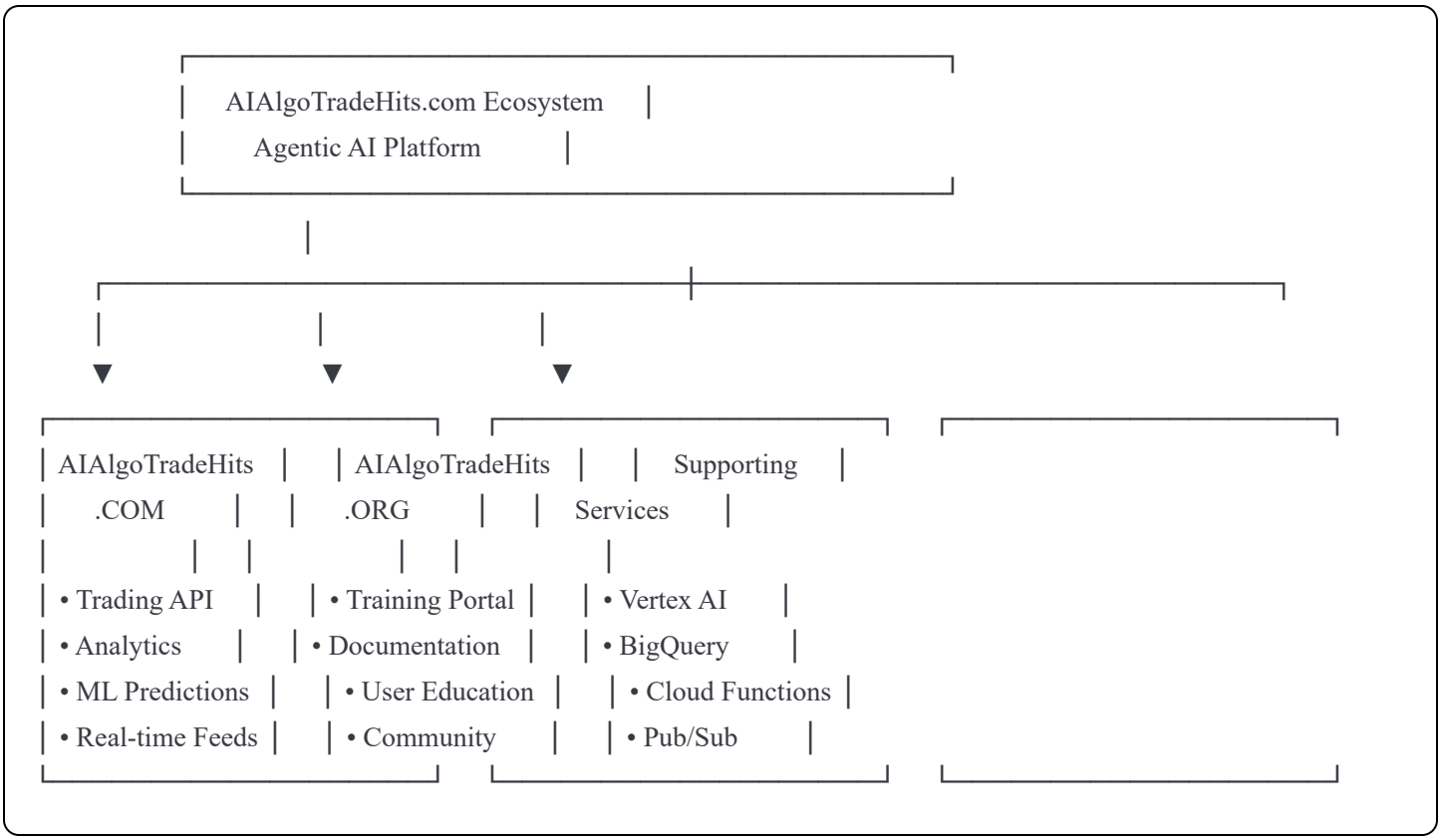
1.2 Architecture Philosophy



1.3 Key Metrics

Metric	Current	Target
Model Accuracy	52.8%	66-72%
API Utilization	0.63%	50%
Prediction Latency	~500ms	<200ms
Daily Data Points	500K+	2M+
Active Agents	7	15

1.4 Platform Ecosystem



---

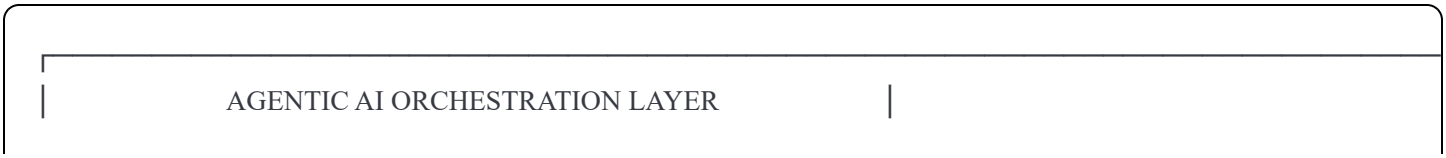
2. AGENTIC AI ARCHITECTURE OVERVIEW

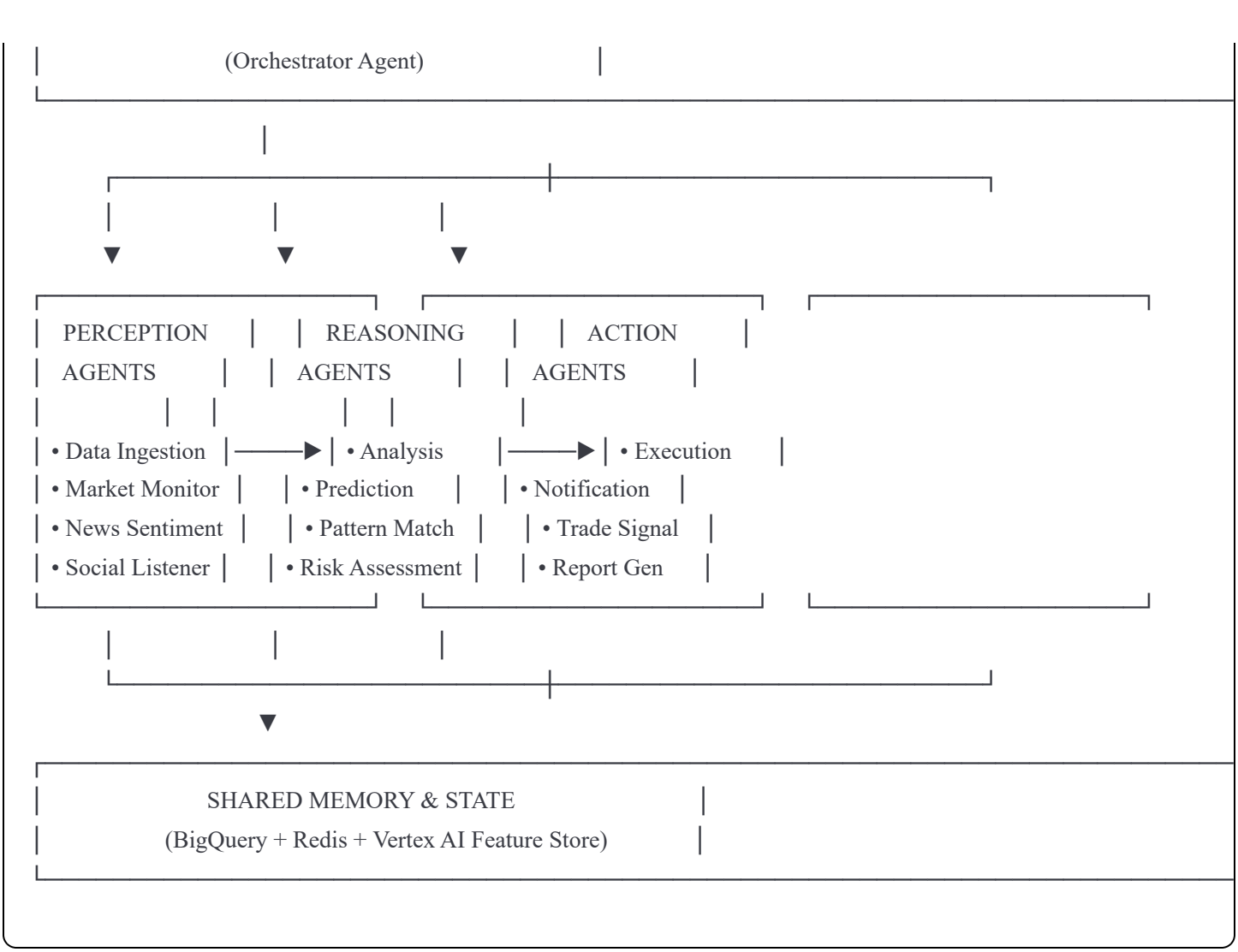
2.1 What is Agentic AI?

Agentic AI represents a paradigm shift from monolithic AI systems to **distributed, autonomous agents** that collaborate to achieve complex goals. Each agent:

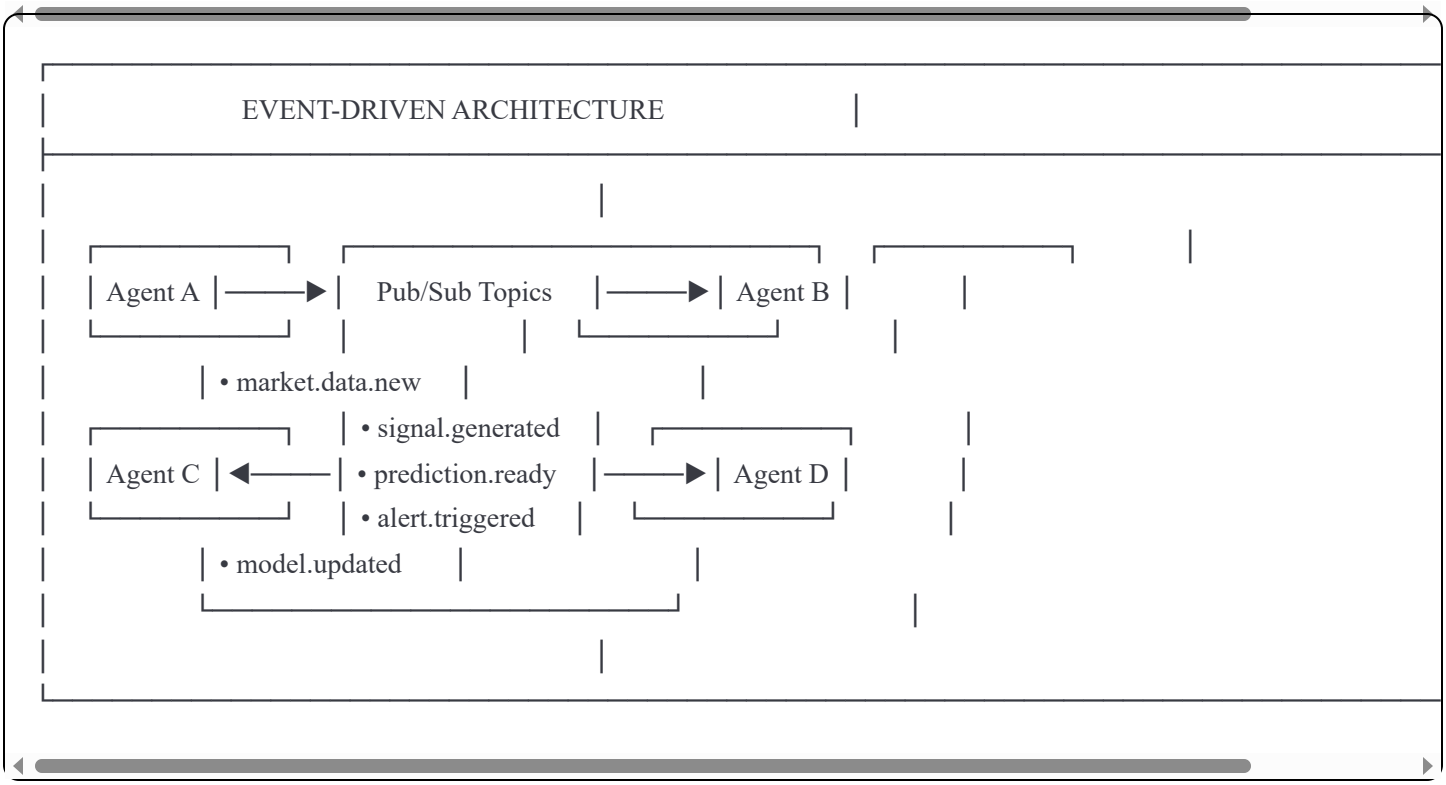
- **Perceives** its environment through sensors and data feeds
- **Reasons** using domain-specific AI models
- **Acts** autonomously within defined boundaries
- **Learns** from outcomes to improve future decisions
- **Communicates** with other agents via standardized protocols

2.2 Core Architecture Pattern

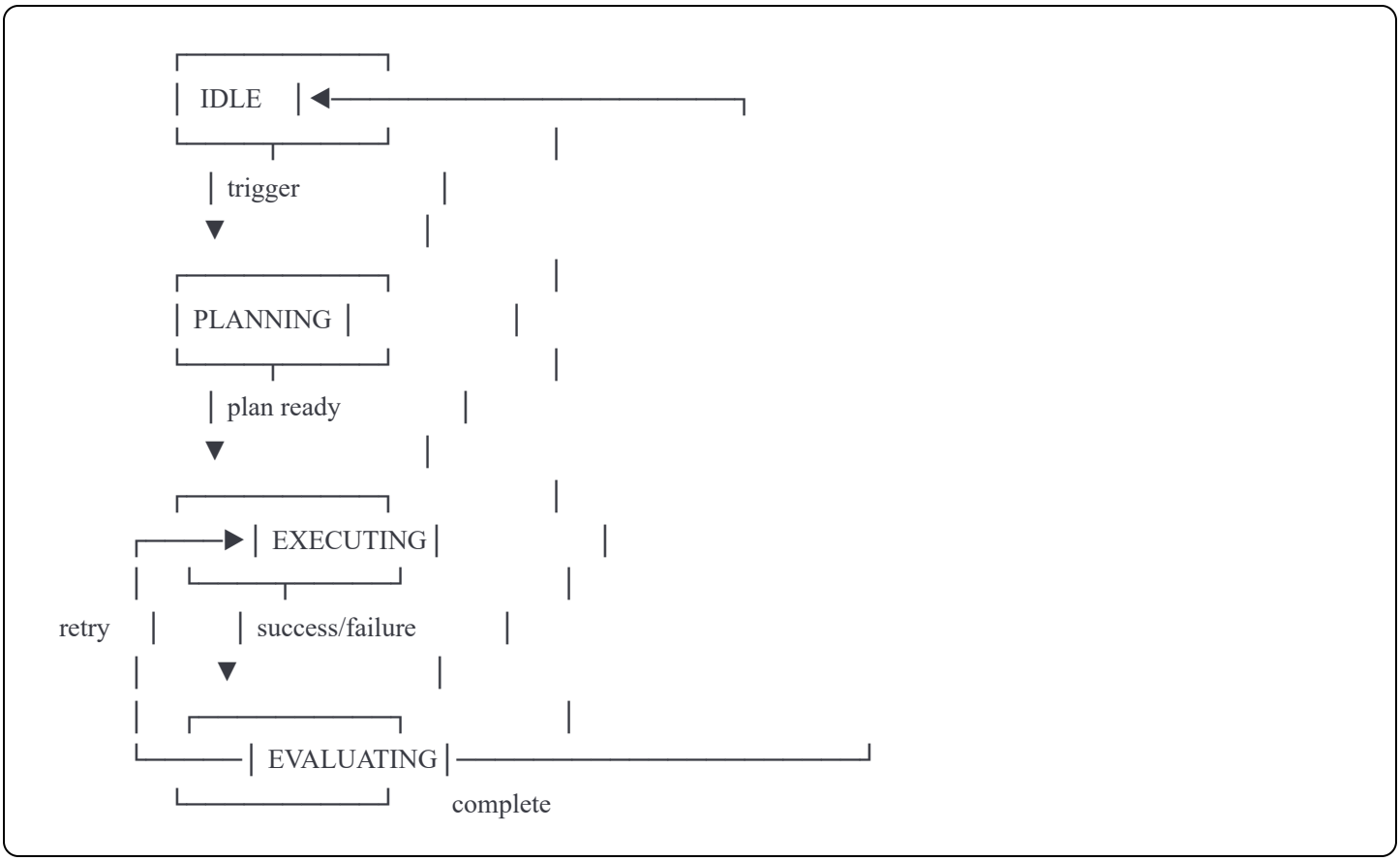




2.3 Agent Communication Model



2.4 Agent Lifecycle States



---

3. AGENT TAXONOMY & DEFINITIONS

3.1 Agent Classification Matrix

AGENT CLASSIFICATION MATRIX			
CATEGORY	AGENT TYPE	AUTONOMY LEVEL	DECISION SCOPE
INFRASTRUCTURE	Data Collector	High	Data Operations
	Health Monitor	High	System State
	Scheduler	High	Task Timing
INTELLIGENCE	Pattern Detector	Medium	Signal Generation
	ML Predictor	Medium	Forecasting
	Sentiment Analyzer	Medium	Market Mood
	Risk Assessor	Medium	Risk Scoring
INTERACTION	NL2SQL Engine	Low	Query Translation

	Report Generator	Low	Document Creation
	Alert Manager	Medium	Notification
EXECUTION	Trade Signal	Low	Signal Routing
	Portfolio Manager	Low	Position Tracking
	Order Router	Low	Execution

3.2 Primary Agent Definitions

3.2.1 ORCHESTRATOR AGENT (Master Controller)

yaml

```yaml
agent_id: orchestrator-001
name: Master Orchestrator Agent
type: COORDINATOR
autonomy_level: HIGH
responsibilities:
  - Coordinate all agent activities
  - Manage workflow execution
  - Handle cross-agent communication
  - Implement circuit breakers
  - Manage global state

triggers:
  - scheduled_jobs
  - api_requests
  - agent_events
  - error_conditions

outputs:
  - workflow_commands
  - agent_directives
  - state_updates

gcp_implementation:
  service: Cloud Run
  url: https://orchestrator-agent-{hash}.run.app
  memory: 2Gi
  cpu: 2
  min_instances: 1
  max_instances: 10

health_check:
  endpoint: /health
  interval: 30s
  timeout: 10s
```

3.2.2 DATA INGESTION AGENT

yaml

```yaml
agent_id: data-ingestion-001
name: Market Data Ingestion Agent
type: PERCEPTION
autonomy_level: HIGH
responsibilities:
  - Fetch OHLCV data from TwelveData, Kraken, CoinMarketCap
  - Normalize data to canonical format
  - Validate data quality
  - Handle rate limits and retries
  - Trigger downstream processing

data_sources:
  - twelvedata:
      api_key: ${TWELVEDATA_API_KEY}
      rate_limit: 800/min
      batch_size: 8
  - kraken:
      ws_url: wss://ws.kraken.com
      rest_url: https://api.kraken.com
  - coinmarketcap:
      api_key: ${CMC_API_KEY}
      rate_limit: 333/day

asset_coverage:
  stocks: 1000+
  crypto: 200+
  etfs: 500+
  forex: 50+
  indices: 30+
  commodities: 20+

outputs:
  - raw_candle_data → BigQuery Bronze Layer
  - data_quality_metrics → Monitoring
  - ingestion_events → Pub/Sub

gcp_implementation:
  service: Cloud Functions (Gen2)
  functions:
    - daily-stock-fetcher
    - daily-crypto-fetcher
    - hourly-stock-fetcher
    - hourly-crypto-fetcher
```

```
  - 5min-crypto-fetcher
  - fundamentals-fetcher

scheduling:
  daily: "0 0 * * *"
  hourly: "0 * * * *"
  5min: "*/5 * * * *"
```

3.2.3 INDICATOR CALCULATION AGENT

yaml

```yaml
agent_id: indicator-calc-001
name: Technical Indicator Calculation Agent
type: PROCESSING
autonomy_level: HIGH
responsibilities:
  - Calculate 24 core technical indicators
  - Generate Growth Score (0-100)
  - Detect EMA crossovers and cycles
  - Classify trend regimes
  - Create ML feature vectors

indicators_by_timeframe:
  daily: 24  # Full indicator set
    - Momentum: RSI, Stochastic, CCI, Williams%R, ROC
    - Trend: EMA(9,21,50,200), SMA(20,50,200), ADX, PSAR
    - Volatility: ATR, Bollinger, Keltner, Donchian
    - Trend_Strength: ADX, Aroon
    - Volume_Flow: OBV, VWAP, A/D, CMF, MFI

  hourly: 12  # Reduced for efficiency
    - RSI, MACD, EMA(9,21,50), ATR, Volume_Ratio
    - Bollinger, ADX, OBV, Stochastic

  5min: 8    # Minimal for speed
    - RSI, MACD, EMA(9,21), ATR, Volume_Ratio, VWAP

  1min: 5    # Ultra-minimal
    - RSI, EMA(9,21), Volume_Ratio, ATR

feature_engineering:
  growth_score:
    formula: weighted_average(momentum_signals, trend_signals, volume_signals)
    range: 0-100
    threshold_buy: 75
    threshold_sell: 25

  trend_regime:
    classes: [STRONG_UP, UP, NEUTRAL, DOWN, STRONG_DOWN]
    features: [ema_slope, adx_value, price_position]

outputs:
  - calculated_indicators → BigQuery Gold Layer
  - feature_vectors → Vertex AI Feature Store
```

```
  - calculation_events → Pub/Sub

gcp_implementation:
  service: Cloud Functions + BigQuery ML
  compute: n1-standard-4
```

3.2.4 ML PREDICTION AGENT

yaml

```yaml
agent_id: ml-prediction-001
name: Machine Learning Prediction Agent
type: REASONING
autonomy_level: MEDIUM
responsibilities:
  - Generate directional predictions (UP/DOWN)
  - Calculate prediction confidence scores
  - Ensemble model coordination
  - Model performance monitoring
  - Feature importance analysis

models:
  primary:
    name: XGBoost Direction Predictor
    model_id: direction_predictor_xgboost
    features: 24
    accuracy: 52.8% (baseline) → 66-72% (target)
    latency: <50ms

  secondary:
    name: Random Forest Ensemble
    accuracy: 62-67%
    latency: <100ms

  advanced:
    name: LSTM Temporal Model
    accuracy: 67-72%
    latency: 100-200ms

  production:
    name: Ensemble Weighted Average
    strategy: weighted_voting
    weights: [0.5, 0.3, 0.2]  # XGB, RF, LSTM
    accuracy: 68-73%

prediction_types:
  - direction_1d: Next day up/down
  - direction_5d: 5-day trend
  - magnitude: Expected % move
  - confidence: Prediction certainty (0-100)

outputs:
  - predictions → BigQuery ml_models.predictions
```

```
  - signals → Trading API
  - performance_metrics → Monitoring

gcp_implementation:
  service: Vertex AI Endpoints
  endpoint_id: prediction-endpoint-001
  machine_type: n1-standard-4
  accelerator: none
  min_replicas: 1
  max_replicas: 5
```

3.2.5 NL2SQL QUERY AGENT (Gemini-Powered)

yaml

```yaml
agent_id: nl2sql-001
name: Natural Language to SQL Query Agent
type: INTERACTION
autonomy_level: LOW
responsibilities:
  - Parse natural language trading queries
  - Generate optimized BigQuery SQL
  - Execute queries safely
  - Format results for users
  - Learn from query patterns

ai_engine:
  model: gemini-2.5-pro
  project_id: aialgotradehits
  location: us-central1
  temperature: 0.1  # Low for accuracy
  max_tokens: 8192

query_types:
  - screening: "Show me oversold cryptos with high volume"
  - analysis: "What's the trend for AAPL?"
  - comparison: "Compare BTC vs ETH momentum"
  - historical: "What patterns preceded the last rally?"
  - portfolio: "Show my exposure to tech stocks"

schema_context:
  tables:
    - crypto_analysis (94K rows)
    - stock_analysis (424K rows)
    - crypto_hourly_data
    - stock_hourly_data
    - ml_models.predictions
    - daily_features_24

safety_constraints:
  - max_rows: 10000
  - max_query_cost: $1.00
  - blocked_operations: [DROP, DELETE, UPDATE, INSERT]
  - rate_limit: 60/min

outputs:
  - sql_queries → BigQuery
  - results → API Response
```

```
  - query_logs → Analytics

gcp_implementation:
  service: Cloud Run
  endpoint: /api/ai/text-to-sql
```

3.2.6 PATTERN RECOGNITION AGENT

yaml

agent_id: pattern-detect-001
name: Chart Pattern Recognition Agent
type: REASONING
autonomy_level: MEDIUM
responsibilities:
  - Detect candlestick patterns
  - Identify chart formations
  - Recognize support/resistance zones
  - Calculate Fibonacci levels
  - Detect Elliott Wave structures

patterns:
  candlestick:
    - Doji, Hammer, Engulfing, Morning Star
    - Evening Star, Three White Soldiers
    - Harami, Marubozu, Spinning Top

  chart_formations:
    - Head and Shoulders
    - Double Top/Bottom
    - Triangle (Ascending, Descending, Symmetric)
    - Flag, Pennant, Wedge
    - Cup and Handle

  fibonacci:
    - Retracement levels: 23.6%, 38.2%, 50%, 61.8%, 78.6%
    - Extension levels: 127.2%, 161.8%, 261.8%

  elliott_wave:
    - Impulse waves (1-5)
    - Corrective waves (A-B-C)

detection_confidence:
  high: >80%
  medium: 60-80%
  low: <60%

outputs:
  - detected_patterns → BigQuery patterns table
  - pattern_alerts → Notification Agent
  - pattern_events → Pub/Sub

```yaml
gcp_implementation:
  service: Cloud Functions + Vision AI (optional)
```

3.2.7 SENTIMENT ANALYSIS AGENT

yaml

```yaml
agent_id: sentiment-001
name: Market Sentiment Analysis Agent
type: PERCEPTION
autonomy_level: MEDIUM
responsibilities:
  - Analyze financial news sentiment
  - Process social media signals
  - Calculate Fear & Greed Index
  - Track institutional sentiment
  - Detect narrative shifts

data_sources:
  - finnhub:
      endpoint: /news
      categories: [general, crypto, forex]
  - social:
      platforms: [twitter, reddit, stocktwits]
  - institutional:
      sources: [13f_filings, insider_trades]

sentiment_scores:
  scale: -100 to +100
  categories:
    - EXTREME_FEAR: -100 to -50
    - FEAR: -50 to -20
    - NEUTRAL: -20 to +20
    - GREED: +20 to +50
    - EXTREME_GREED: +50 to +100

analysis_methods:
  - NLP: Gemini 2.5 Pro sentiment extraction
  - Lexicon: FinBERT financial sentiment
  - Volume: Social mention velocity
  - Contrast: Sentiment vs price divergence

outputs:
  - sentiment_scores → BigQuery sentiment table
  - sentiment_alerts → Notification Agent
  - sentiment_events → Pub/Sub

gcp_implementation:
```

service: Cloud Functions
ai_model: Gemini 2.5 Pro + FinBERT

3.2.8 ALERT & NOTIFICATION AGENT

yaml

agent_id: alert-001
name: Alert & Notification Management Agent
type: ACTION
autonomy_level: MEDIUM
responsibilities:
  - Monitor alert conditions
  - Deduplicate notifications
  - Route alerts to channels
  - Track acknowledgments
  - Manage alert fatigue

alert_types:
  - price_alert: Target price reached
  - pattern_alert: Pattern detected
  - signal_alert: Trading signal generated
  - risk_alert: Risk threshold exceeded
  - system_alert: System health issues

channels:
  - email: SendGrid API
  - push: Firebase Cloud Messaging
  - sms: Twilio (critical only)
  - webhook: Custom integrations
  - dashboard: Real-time UI updates

priority_levels:
  - P1_CRITICAL: Immediate action required
  - P2_HIGH: Act within 1 hour
  - P3_MEDIUM: Daily review
  - P4_LOW: Weekly digest

anti_fatigue:
  - deduplication_window: 5min
  - max_alerts_per_hour: 20
  - quiet_hours: configurable
  - escalation_rules: defined

outputs:
  - notifications → Delivery channels
  - alert_logs → BigQuery audit table
  - delivery_metrics → Monitoring

```yaml
gcp_implementation:
  service: Cloud Functions + Cloud Tasks
```

3.2.9 REPORT GENERATION AGENT

yaml

```yaml
agent_id: report-001
name: AI Report Generation Agent
type: ACTION
autonomy_level: LOW
responsibilities:
  - Generate daily market summaries
  - Create portfolio performance reports
  - Build custom analysis documents
  - Produce compliance reports
  - Generate training data documentation

report_types:
  daily_market_summary:
    content: [top_movers, sector_performance, signals, sentiment]
    format: [PDF, HTML, Email]
    schedule: "0 7 * * 1-5"

  portfolio_performance:
    content: [positions, pnl, risk_metrics, attribution]
    format: [PDF, XLSX]
    schedule: on_demand

  signal_analysis:
    content: [signals, accuracy, backtest_results]
    format: [PDF, HTML]
    schedule: weekly

  compliance:
    content: [trades, positions, exposure, alerts]
    format: [PDF]
    schedule: monthly

ai_enhancement:
  model: Gemini 2.5 Pro
  features:
    - Natural language market commentary
    - Trend interpretation
    - Risk explanation
    - Recommendation synthesis

outputs:
  - reports → Cloud Storage
  - report_metadata → BigQuery
```

```
  - delivery_notifications → Alert Agent

gcp_implementation:
  service: Cloud Functions + Document AI
```

3.2.10 HEALTH MONITORING AGENT

yaml

```yaml
agent_id: health-001
name: System Health Monitoring Agent
type: INFRASTRUCTURE
autonomy_level: HIGH
responsibilities:
  - Monitor all agent health
  - Track API quotas and limits
  - Detect anomalies
  - Trigger self-healing
  - Report system status

monitoring_targets:
  agents:
    - All registered agents
    - Health check endpoints
    - Response latencies
    - Error rates

  infrastructure:
    - Cloud Run services
    - Cloud Functions
    - BigQuery slots
    - Pub/Sub throughput

  external_apis:
    - TwelveData quota: 1,152,000/day
    - Kraken rate limits
    - CoinMarketCap credits
    - Vertex AI quotas

health_metrics:
  - availability: 99.9% target
  - latency_p50: <200ms
  - latency_p99: <1000ms
  - error_rate: <0.1%

self_healing:
  - restart_unhealthy: enabled
  - circuit_breaker: enabled
  - fallback_routes: configured
  - auto_scaling: enabled

outputs:
```

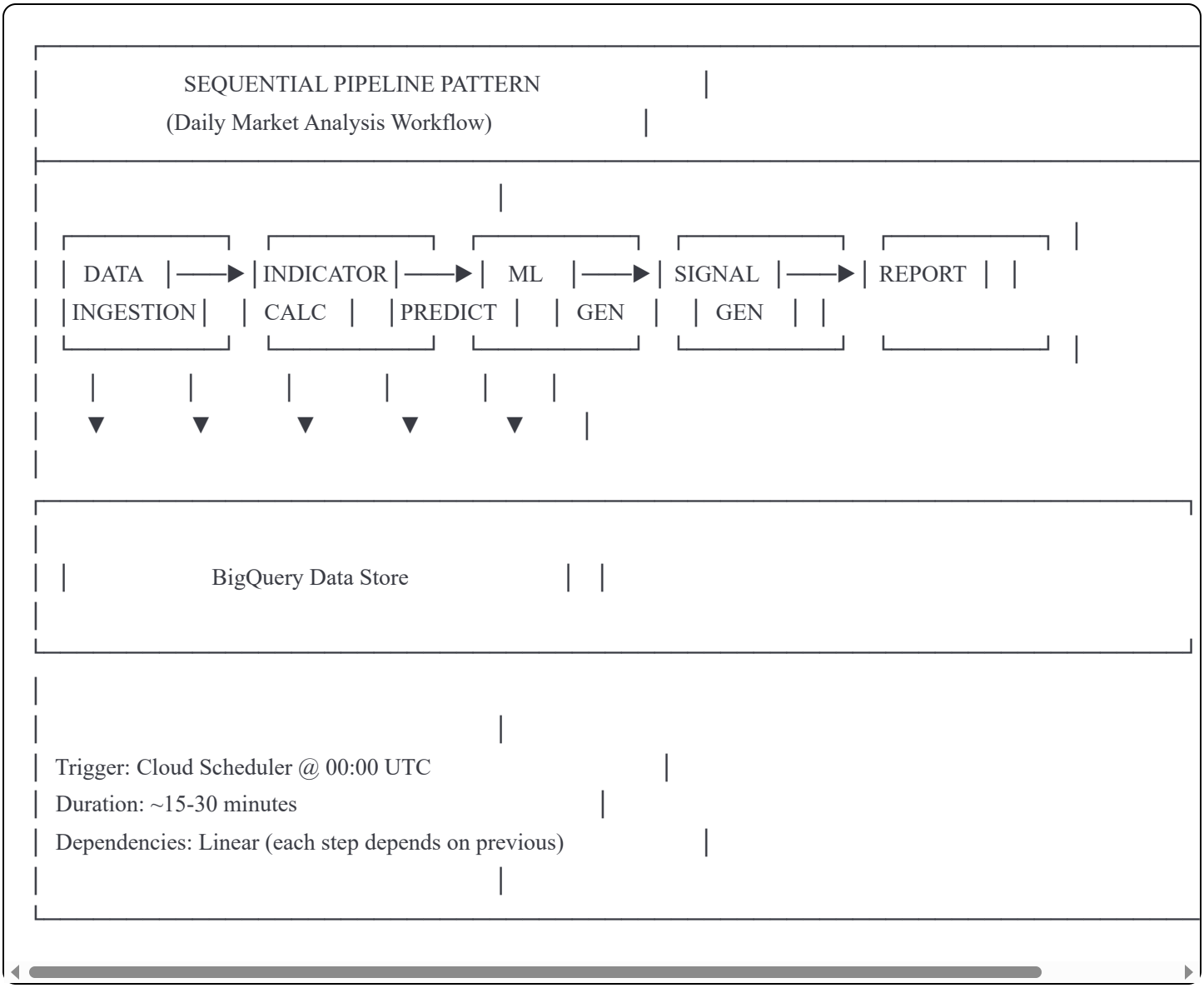
```
- health_status → Dashboard
- alerts → Alert Agent
- metrics → Cloud Monitoring

gcp_implementation:
  service: Cloud Monitoring + Cloud Run
```

---

4. MULTI-AGENT ORCHESTRATION FRAMEWORK

4.1 Orchestration Patterns

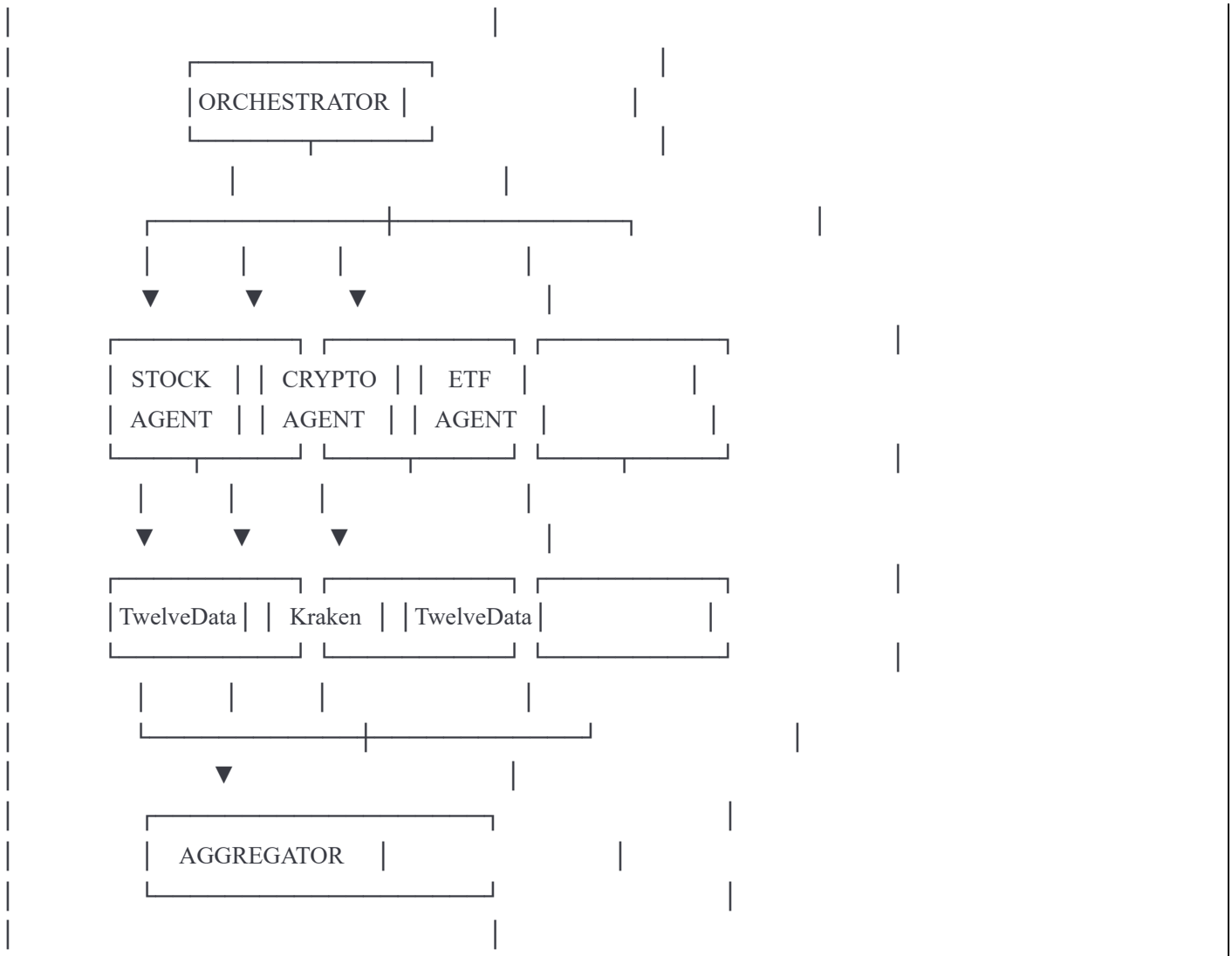
4.1.1 Sequential Pipeline Pattern




```
python
```

```python
# Sequential Pipeline Implementation
class DailyAnalysisPipeline:
    def __init__(self):
        self.orchestrator = OrchestratorAgent()
        self.agents = {
            'ingestion': DataIngestionAgent(),
            'indicator': IndicatorCalcAgent(),
            'prediction': MLPredictionAgent(),
            'signal': SignalGenerationAgent(),
            'report': ReportGenerationAgent()
        }

    async def execute(self, date: str):
        context = {'date': date, 'status': 'started'}

        # Step 1: Data Ingestion
        context = await self.agents['ingestion'].run(context)
        if context['status'] == 'error':
            return await self.handle_error(context)

        # Step 2: Indicator Calculation
        context = await self.agents['indicator'].run(context)
        if context['status'] == 'error':
            return await self.handle_error(context)

        # Step 3: ML Predictions
        context = await self.agents['prediction'].run(context)

        # Step 4: Signal Generation
        context = await self.agents['signal'].run(context)

        # Step 5: Report Generation
        context = await self.agents['report'].run(context)

        return context
```

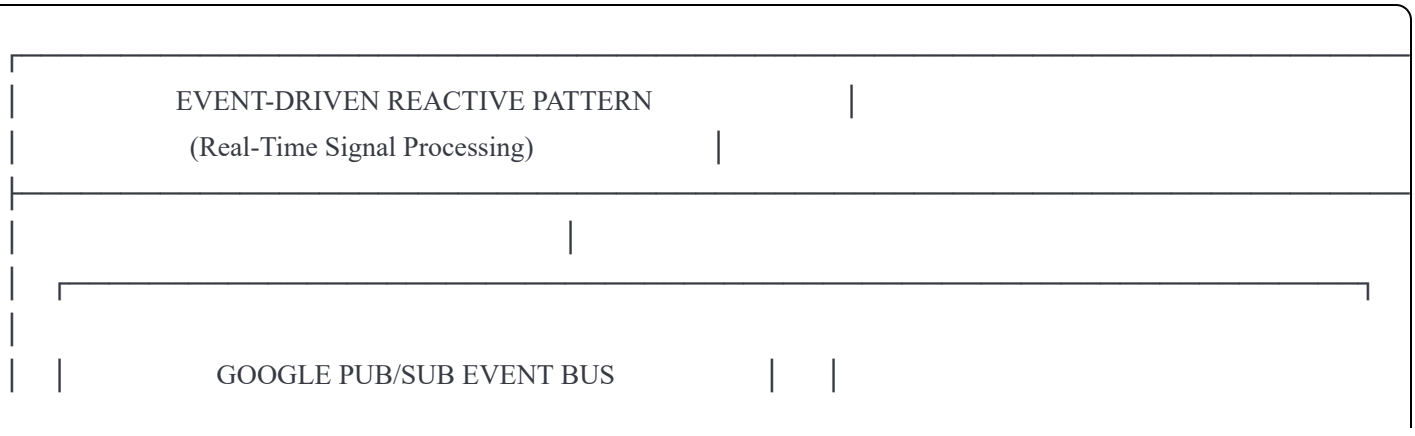
4.1.2 Parallel Fan-Out Pattern

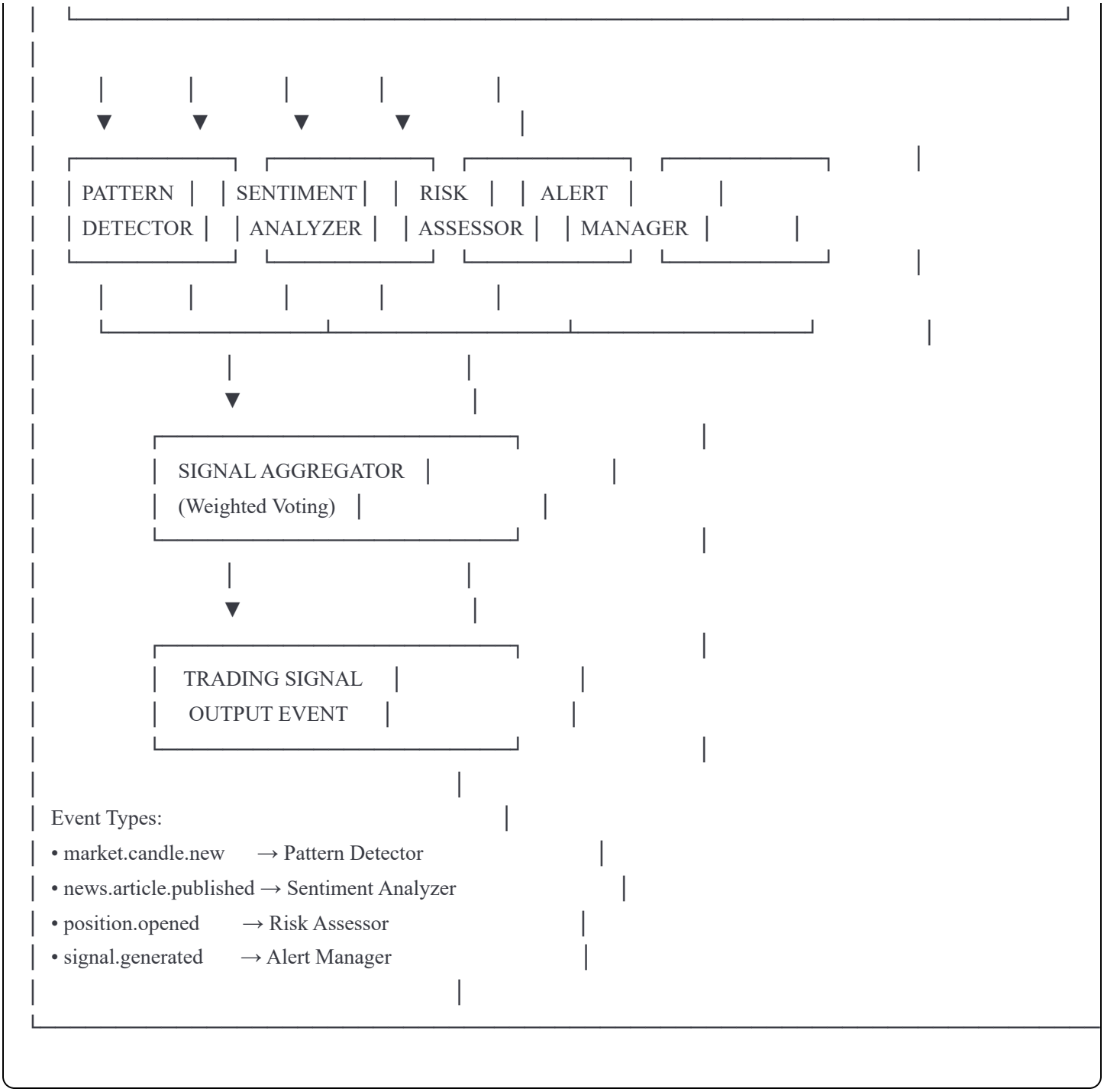

ORCHESTRATOR
STOCK AGENT
CRYPTO AGENT
ETF AGENT
TwelveData
Kraken
TwelveData
AGGREGATOR
Execution: Parallel with 20 workers
Timeout: 5 minutes per agent
Aggregation: Wait for all or timeout

python

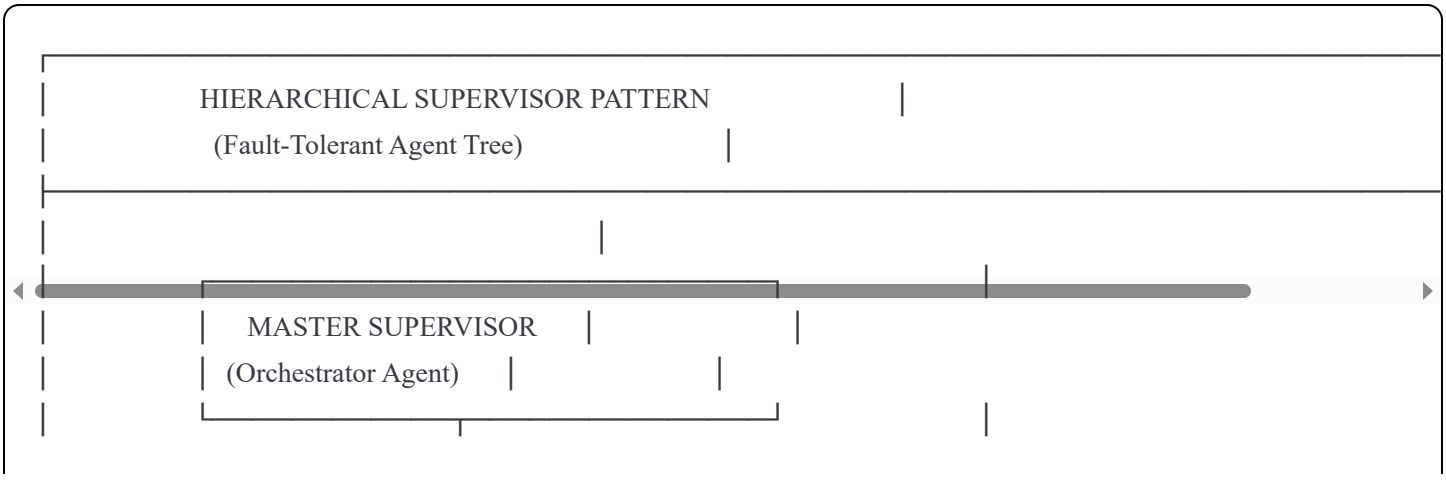
```python
# Parallel Fan-Out Implementation
class ParallelDataCollection:
    def __init__(self, max_workers: int = 20):
        self.max_workers = max_workers
        self.agents = {
            'stocks': StockDataAgent(),
            'crypto': CryptoDataAgent(),
            'etfs': ETFDataAgent(),
            'forex': ForexDataAgent(),
            'indices': IndicesDataAgent(),
            'commodities': CommoditiesDataAgent()
        }

    async def collect_all(self):
        tasks = []
        for name, agent in self.agents.items():
            task = asyncio.create_task(
                agent.collect_with_timeout(timeout=300)
            )
            tasks.append((name, task))

        results = {}
        for name, task in tasks:
            try:
                results[name] = await task
            except asyncio.TimeoutError:
                results[name] = {'status': 'timeout'}
            except Exception as e:
                results[name] = {'status': 'error', 'error': str(e)}

        return await self.aggregate_results(results)
```

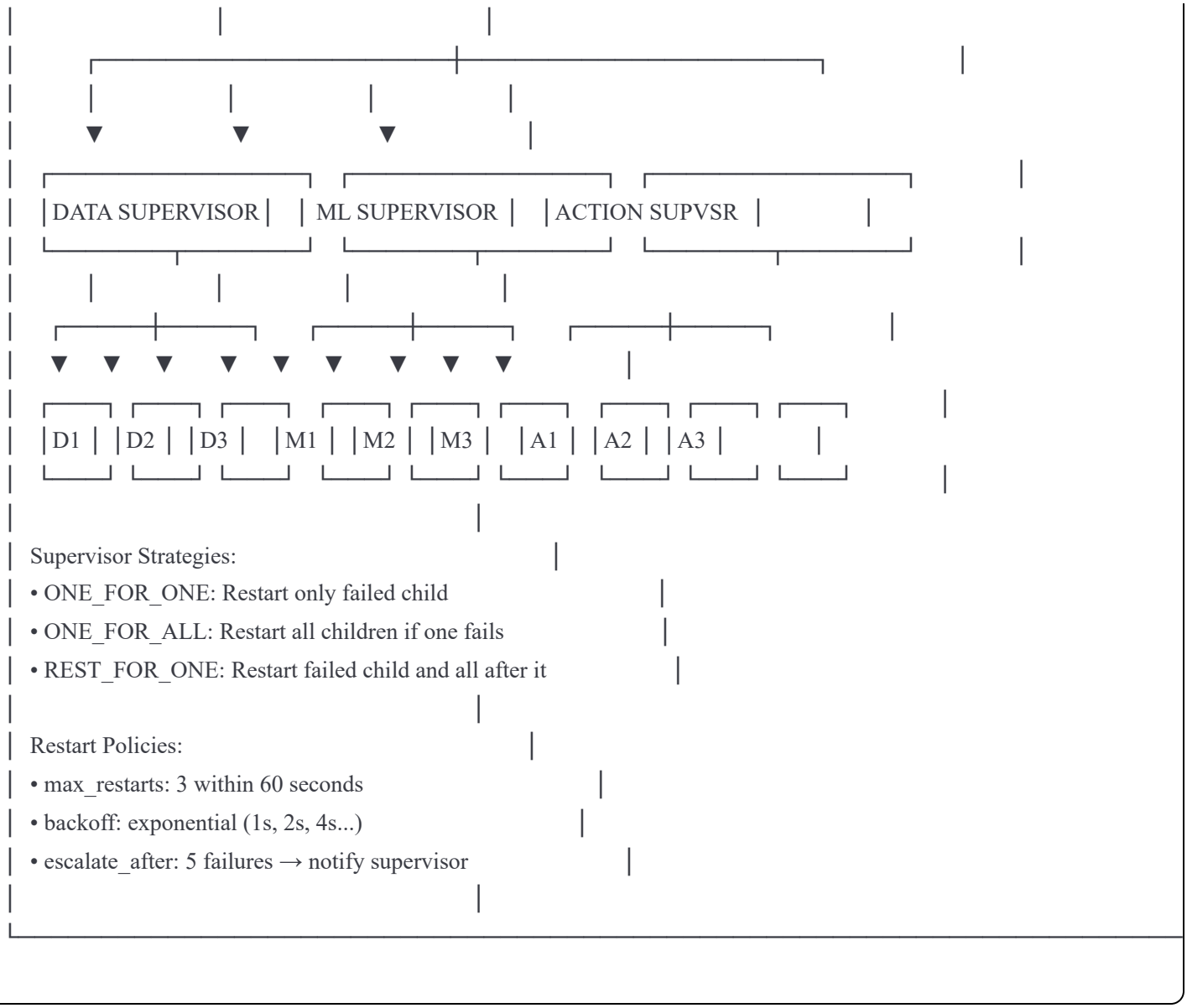
4.1.3 Event-Driven Reactive Pattern





4.1.4 Hierarchical Supervisor Pattern





4.2 Agent Communication Protocol

4.2.1 Message Schema

json

```
{
  "message_id": "uuid-v4",
  "timestamp": "2026-01-03T12:00:00Z",
  "source_agent": "data-ingestion-001",
  "target_agent": "indicator-calc-001",
  "message_type": "TASK_REQUEST",
  "priority": "HIGH",
  "correlation_id": "workflow-123",
  "payload": {
    "task_type": "CALCULATE_INDICATORS",
    "parameters": {
      "symbols": ["AAPL", "BTCUSD", "SPY"],
      "timeframe": "daily",
      "date": "2026-01-03"
    }
  },
  "metadata": {
    "retry_count": 0,
    "max_retries": 3,
    "timeout_ms": 30000,
    "trace_id": "trace-abc123"
  }
}
```


4.2.2 Message Types

yaml

```yaml
message_types:
  TASK_REQUEST:
    description: Request agent to perform a task
    requires_response: true
    timeout: 30s

  TASK_RESPONSE:
    description: Response to a task request
    payload: [result, error, metadata]

  EVENT_NOTIFICATION:
    description: Broadcast event to subscribers
    requires_response: false
    delivery: at_least_once

  STATE_UPDATE:
    description: Agent state change notification
    broadcast: true

  HEALTH_CHECK:
    description: Periodic health verification
    interval: 30s

  COMMAND:
    description: Direct command to agent
    authority: orchestrator_only
```

4.3 Workflow Definitions

4.3.1 Daily Data Collection Workflow

yaml

```yaml
workflow_id: daily-data-collection
name: Daily Market Data Collection
schedule: "0 0 * * *"  # Midnight UTC
timeout: 1800s  # 30 minutes

steps:
  - step_id: 1
    agent: data-ingestion-001
    action: fetch_daily_data
    inputs:
      asset_classes: [stocks, crypto, etfs, forex, indices, commodities]
      date: ${workflow.date}
    outputs:
      - raw_data_count
      - data_quality_score
    on_error: retry(3, exponential)

  - step_id: 2
    agent: indicator-calc-001
    action: calculate_all_indicators
    depends_on: [1]
    inputs:
      data_date: ${step_1.date}
      indicator_set: full_24
    outputs:
      - indicators_calculated
      - features_stored

  - step_id: 3
    agent: ml-prediction-001
    action: generate_predictions
    depends_on: [2]
    inputs:
      feature_date: ${step_2.date}
      models: [xgboost, ensemble]
    outputs:
      - predictions
      - confidence_scores

  - step_id: 4
    agent: pattern-detect-001
    action: scan_patterns
    depends_on: [2]
```

```yaml
    parallel: true
    inputs:
      scan_date: ${step_2.date}
      patterns: all

  - step_id: 5
    agent: report-001
    action: generate_daily_summary
    depends_on: [3, 4]
    inputs:
      predictions: ${step_3.predictions}
      patterns: ${step_4.patterns}

notifications:
  on_success:
    - type: email
      to: admin@aialgotradehits.com
      template: daily_success
  on_failure:
    - type: pagerduty
      severity: high
```

4.3.2 Real-Time Signal Generation Workflow

yaml

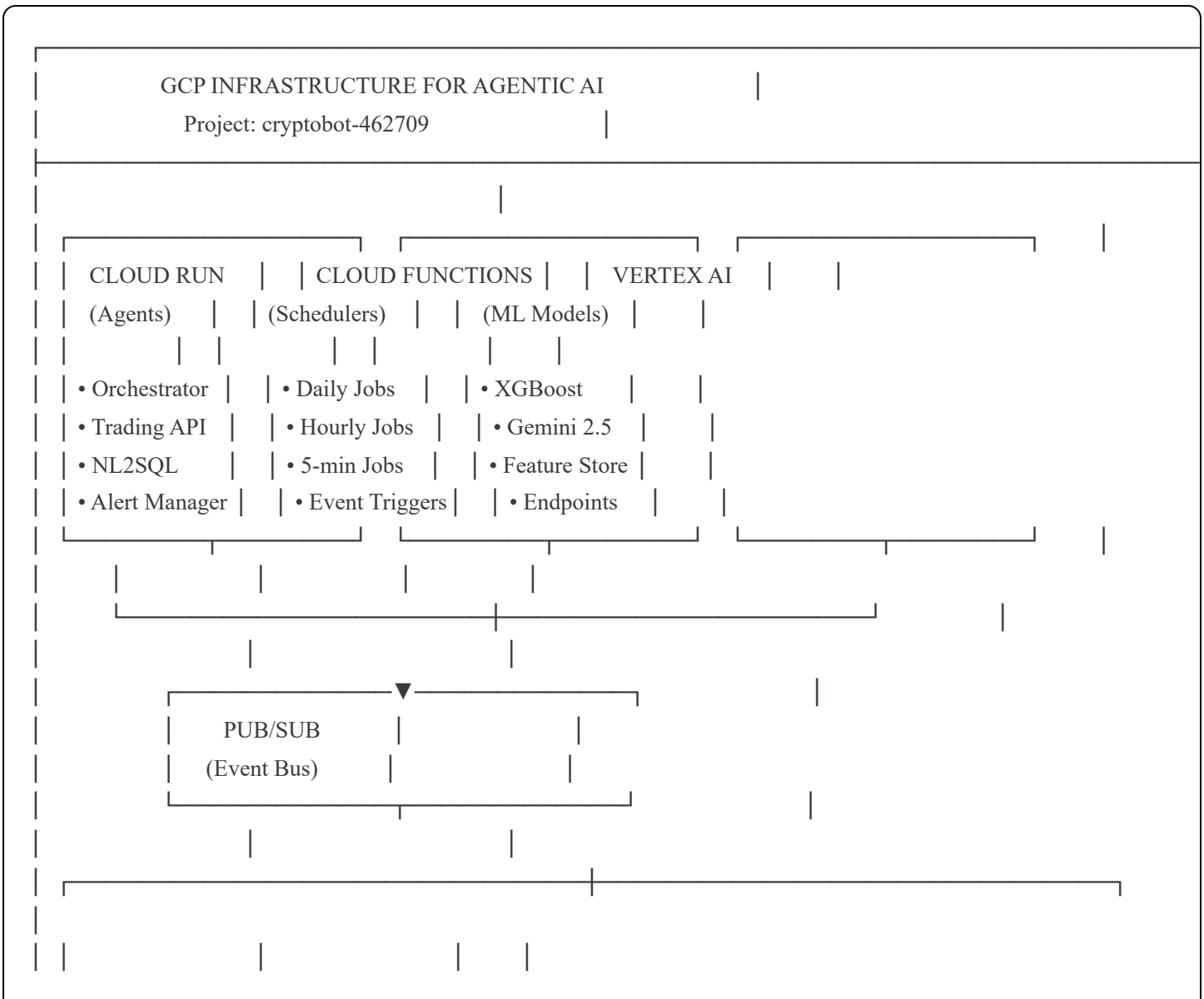
```yaml
workflow_id: realtime-signal-generation
name: Real-Time Trading Signal Generation
trigger: event
event_source: market.candle.new

steps:
  - step_id: 1
    agent: indicator-calc-001
    action: update_indicators
    inputs:
      symbol: ${event.symbol}
      candle: ${event.candle}
    timeout: 5s

  - step_id: 2
    agent: pattern-detect-001
    action: detect_patterns
    depends_on: [1]
    timeout: 3s

  - step_id: 3
    agent: ml-prediction-001
    action: quick_predict
    depends_on: [1]
    timeout: 2s

  - step_id: 4
    agent: sentiment-001
    action: get_current_sentiment
    parallel_with: [2, 3]
    timeout: 3s

  - step_id: 5
    agent: signal-generator-001
    action: aggregate_signals
    depends_on: [2, 3, 4]
    inputs:
      pattern_signal: ${step_2.signal}
      ml_signal: ${step_3.signal}
      sentiment_signal: ${step_4.signal}
    outputs:
      - final_signal
      - confidence
```

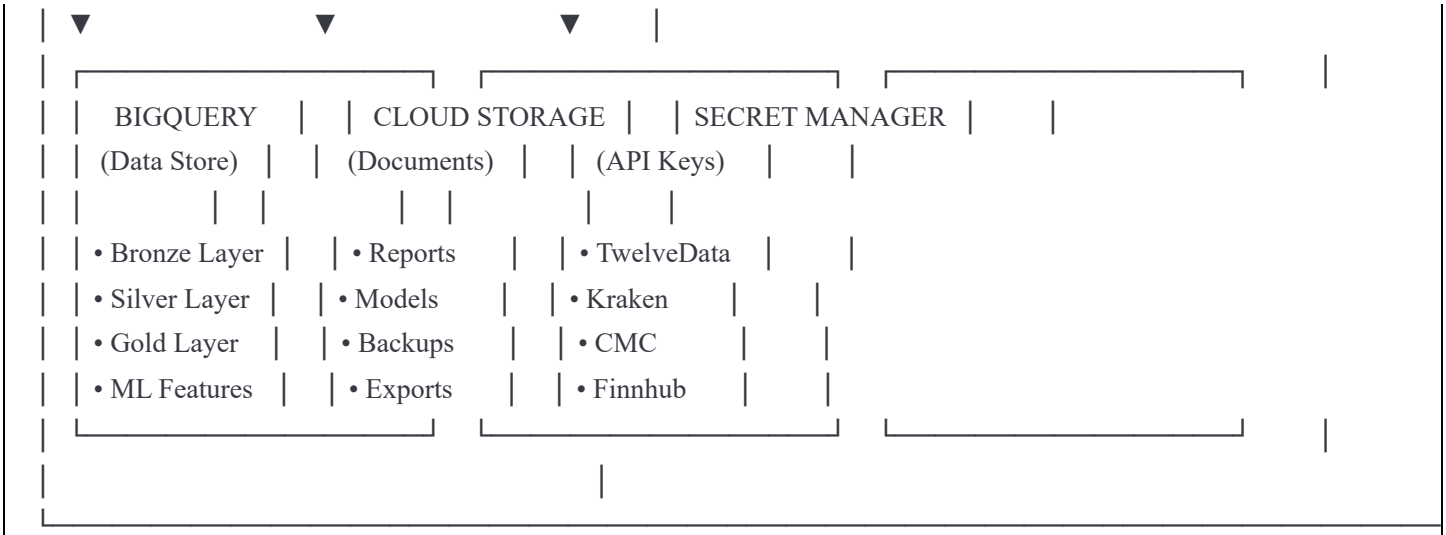
```
  - step_id: 6
    agent: alert-001
    action: send_notification
    condition: ${step_5.confidence} > 0.7
    depends_on: [5]

sla:
  target_latency_p50: 500ms
  target_latency_p99: 2000ms
```

---

5. GCP INFRASTRUCTURE INTEGRATION

5.1 Infrastructure Overview





5.2 Service Specifications

5.2.1 Cloud Run Services

yaml

```yaml
services:
  orchestrator-agent:
    name: orchestrator-agent
    region: us-central1
    cpu: 2
    memory: 2Gi
    min_instances: 1
    max_instances: 10
    concurrency: 80
    timeout: 300s
    env_vars:
      - PROJECT_ID: cryptobot-462709
      - PUBSUB_TOPIC: agent-events
    secrets:
      - TWELVEDATA_API_KEY
      - ANTHROPIC_API_KEY

  trading-api:
    name: trading-api
    region: us-central1
    cpu: 2
    memory: 2Gi
    min_instances: 1
    max_instances: 20
    concurrency: 100
    timeout: 60s
    url: https://trading-api-1075463475276.us-central1.run.app

  nl2sql-agent:
    name: nl2sql-agent
    region: us-central1
    cpu: 1
    memory: 1Gi
    min_instances: 0
    max_instances: 5
    concurrency: 20
    timeout: 30s
```

5.2.2 Cloud Functions (Gen2)

yaml

```yaml
functions:
  daily-stock-fetcher:
    runtime: python311
    memory: 512MB
    timeout: 540s
    trigger: Cloud Scheduler
    schedule: "0 0 * * *"
    entry_point: fetch_daily_stocks
    max_instances: 10

  daily-crypto-fetcher:
    runtime: python311
    memory: 512MB
    timeout: 540s
    trigger: Cloud Scheduler
    schedule: "0 0 * * *"
    entry_point: fetch_daily_crypto

  hourly-data-fetcher:
    runtime: python311
    memory: 256MB
    timeout: 300s
    trigger: Cloud Scheduler
    schedule: "0 * * * *"
    entry_point: fetch_hourly_data

  5min-crypto-fetcher:
    runtime: python311
    memory: 256MB
    timeout: 180s
    trigger: Cloud Scheduler
    schedule: "*/5 * * * *"
    entry_point: fetch_5min_crypto

  indicator-calculator:
    runtime: python311
    memory: 1GB
    timeout: 540s
    trigger: Pub/Sub
    topic: candle-ingested
    entry_point: calculate_indicators

  pattern-detector:
```

```
  runtime: python311
  memory: 512MB
  timeout: 300s
  trigger: Pub/Sub
  topic: indicators-calculated
  entry_point: detect_patterns

signal-generator:
  runtime: python311
  memory: 512MB
  timeout: 60s
  trigger: Pub/Sub
  topic: analysis-complete
  entry_point: generate_signals
```

5.2.3 Cloud Scheduler Jobs

yaml

```yaml
schedulers:
  - name: daily-crypto-fetch-job
    schedule: "0 0 * * *"
    timezone: UTC
    target:
      type: http
      uri: https://us-central1-cryptobot-462709.cloudfunctions.net/daily-crypto-fetcher

  - name: daily-stock-fetch-job
    schedule: "0 1 * * 1-5"  # Weekdays only
    timezone: America/New_York
    target:
      type: http
      uri: https://us-central1-cryptobot-462709.cloudfunctions.net/daily-stock-fetcher

  - name: hourly-crypto-fetch-job
    schedule: "0 * * * *"
    target:
      type: http
      uri: https://us-central1-cryptobot-462709.cloudfunctions.net/hourly-data-fetcher

  - name: 5min-top10-fetch-job
    schedule: "*/5 * * * *"
    target:
      type: http
      uri: https://us-central1-cryptobot-462709.cloudfunctions.net/5min-crypto-fetcher

  - name: ml-model-retrain-job
    schedule: "0 2 * * 0"  # Weekly Sunday 2 AM
    target:
      type: http
      uri: https://us-central1-cryptobot-462709.cloudfunctions.net/ml-model-trainer
```

5.2.4 Pub/Sub Topics & Subscriptions

yaml

```yaml
topics:
  candle-ingested:
    description: New candle data has been stored
    retention: 7d
    subscriptions:
      - indicator-calculator-sub
      - data-quality-monitor-sub

  indicators-calculated:
    description: Indicators have been calculated for a symbol
    retention: 7d
    subscriptions:
      - pattern-detector-sub
      - ml-predictor-sub

  pattern-detected:
    description: Chart pattern has been detected
    retention: 7d
    subscriptions:
      - signal-generator-sub
      - alert-manager-sub

  prediction-generated:
    description: ML prediction has been generated
    retention: 7d
    subscriptions:
      - signal-generator-sub
      - dashboard-update-sub

  signal-generated:
    description: Trading signal has been generated
    retention: 30d
    subscriptions:
      - alert-manager-sub
      - audit-logger-sub
      - portfolio-manager-sub

  agent-events:
    description: General agent communication channel
    retention: 1d
    subscriptions:
```

```
    - orchestrator-sub
    - health-monitor-sub
```

5.3 Vertex AI Configuration

yaml

```yaml
vertex_ai:
  project_id: cryptobot-462709
  location: us-central1

  models:
    xgboost_predictor:
      model_id: direction_predictor_xgboost
      artifact_uri: gs://aialgotradehits-models/xgboost/
      container: us-docker.pkg.dev/vertex-ai/prediction/xgboost-cpu.1-6:latest

    gemini_analyzer:
      model_id: gemini-2.5-pro
      endpoint: projects/cryptobot-462709/locations/us-central1/publishers/google/models/gemini-2.5-pro
      temperature: 0.1
      max_tokens: 8192

  endpoints:
    prediction_endpoint:
      display_name: trading-prediction-endpoint
      machine_type: n1-standard-4
      min_replicas: 1
      max_replicas: 5
      traffic_split:
        deployed_model_id: 100

  feature_store:
    featurestore_id: trading_features
    entity_types:
      - symbol_daily_features
      - symbol_hourly_features
    online_serving:
      enabled: true
      fixed_node_count: 1
```

6. PROJECT REGISTRY & AGENT MAPPING

6.1 Complete Project Registry

yaml

```yaml
projects:
  aialgotradehits_com:
    name: AIAlgoTradeHits.com (Primary Trading Platform)
    description: Main trading analytics and prediction platform
    status: PRODUCTION
    url: https://trading-api-1075463475276.us-central1.run.app
    agents:
      - orchestrator-001
      - data-ingestion-001
      - indicator-calc-001
      - ml-prediction-001
      - nl2sql-001
      - pattern-detect-001
      - alert-001
      - health-001
    databases:
      - bigquery: aialgotradehits.trading_data
      - bigquery: aialgotradehits.ml_models

  aialgotradehits_org:
    name: AIAlgoTradeHits.org (Training & Education)
    description: Fintech training and user interaction portal
    status: DEVELOPMENT
    agents:
      - report-001
      - training-content-agent
      - user-interaction-agent
    databases:
      - bigquery: aialgotradehits.training_data

  data_warehouse:
    name: Historical Data Warehouse
    description: Comprehensive historical market data storage
    status: PRODUCTION
    agents:
      - data-ingestion-001
      - data-quality-agent
      - backfill-agent
    databases:
      - bigquery: aialgotradehits.bronze_layer
      - bigquery: aialgotradehits.silver_layer
      - bigquery: aialgotradehits.gold_layer
```

```yaml
ml_training_pipeline:
  name: ML Model Training Pipeline
  description: Automated model training and deployment
  status: PRODUCTION
  agents:
    - ml-training-agent
    - model-evaluation-agent
    - model-deployment-agent
  compute:
    - vertex_ai_training
    - vertex_ai_endpoints

real_time_analytics:
  name: Real-Time Analytics Engine
  description: Sub-second market analysis and alerting
  status: PRODUCTION
  agents:
    - stream-processor-agent
    - real-time-indicator-agent
    - instant-alert-agent
  infrastructure:
    - pubsub
    - dataflow

sentiment_engine:
  name: Sentiment Analysis Engine
  description: News and social sentiment processing
  status: DEVELOPMENT
  agents:
    - sentiment-001
    - news-processor-agent
    - social-listener-agent
  data_sources:
    - finnhub
    - twitter_api
    - reddit_api

backtesting_simulator:
  name: Strategy Backtesting Simulator
  description: Historical strategy validation
  status: PLANNED
  agents:
    - backtest-engine-agent
    - performance-analyzer-agent
```

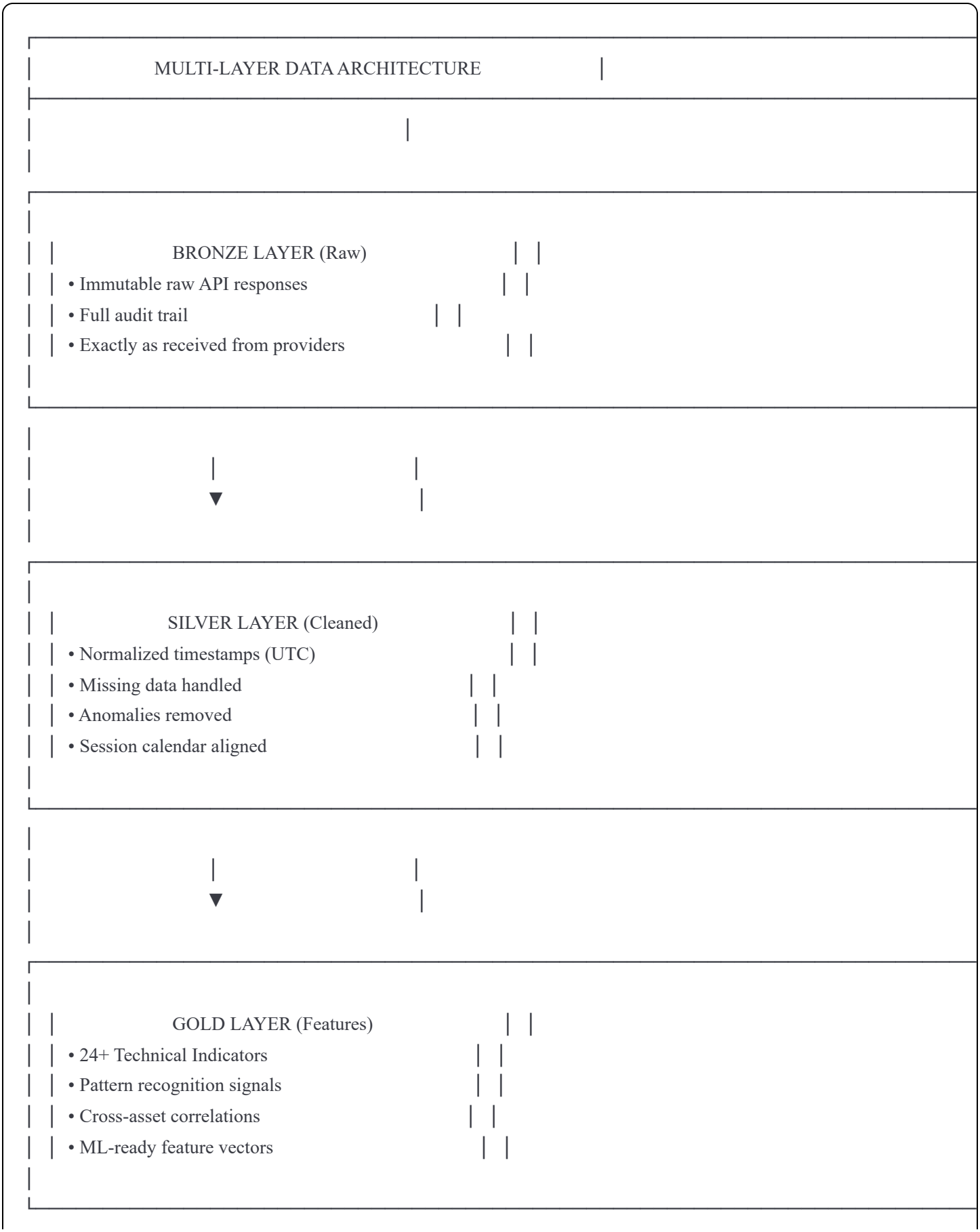
```
    - optimization-agent

  trading_execution:
    name: Trading Execution System
    description: Order routing and execution (future)
    status: PLANNED
    agents:
      - order-router-agent
      - execution-agent
      - position-manager-agent
    exchanges:
      - kraken_pro
      - alpaca
```

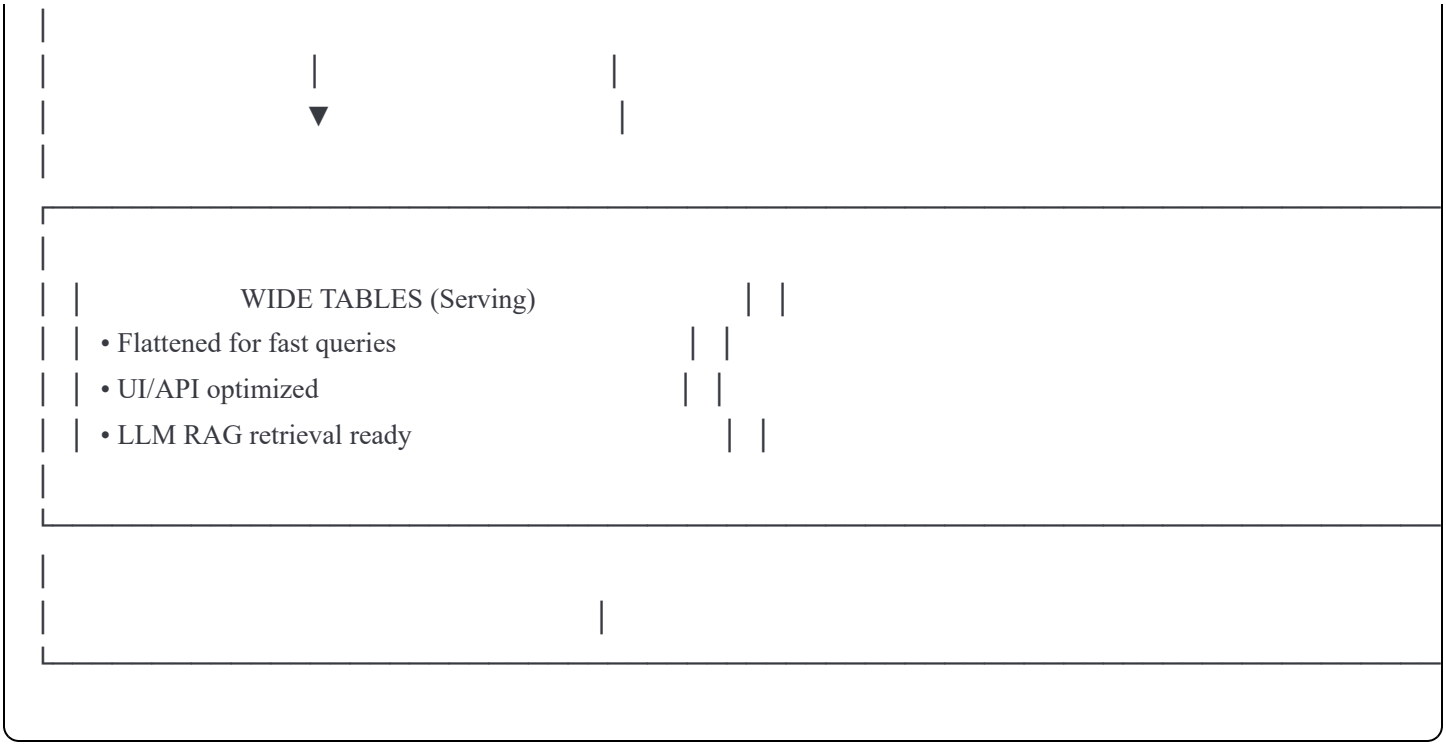
6.2 Agent-to-Project Mapping Matrix

```
┌──────────────────────────────────────────────────────────────────────────────┐
│                    AGENT-TO-PROJECT MAPPING MATRIX                           │
├──────────────────┬──────┬──────┬──────────┬──────────┬──────────┬──────┤
│ AGENT            │ .COM │ .ORG │ WAREHOUSE│ ML PIPE  │ REALTIME │ SENT │
├──────────────────┼──────┼──────┼──────────┼──────────┼──────────┼──────┤
│ orchestrator-001 │  ●   │  ○   │    ○     │    ○     │    ●     │  ○   │
│ data-ingestion-001│  ●   │      │    ●     │          │    ●     │      │
│ indicator-calc-001│  ●   │      │    ●     │    ○     │    ●     │      │
│ ml-prediction-001 │  ●   │  ○   │          │    ●     │    ●     │      │
│ nl2sql-001       │  ●   │  ●   │    ○     │          │          │      │
│ pattern-detect-001│  ●   │  ○   │          │    ○     │    ●     │      │
│ sentiment-001    │  ●   │      │    ○     │    ○     │    ●     │  ●   │
│ alert-001        │  ●   │  ○   │          │          │    ●     │  ○   │
│ report-001       │  ●   │  ●   │    ○     │    ○     │          │      │
│ health-001       │  ●   │  ●   │    ●     │    ●     │    ●     │  ●   │
├──────────────────┴──────┴──────┴──────────┴──────────┴──────────┴──────┤
│ ● = Primary Assignment   ○ = Secondary/Support                          │
└──────────────────────────────────────────────────────────────────────────────┘
```

7. DATA LAYER ARCHITECTURE

7.1 Multi-Layer Data Architecture





7.2 BigQuery Table Schema

sql

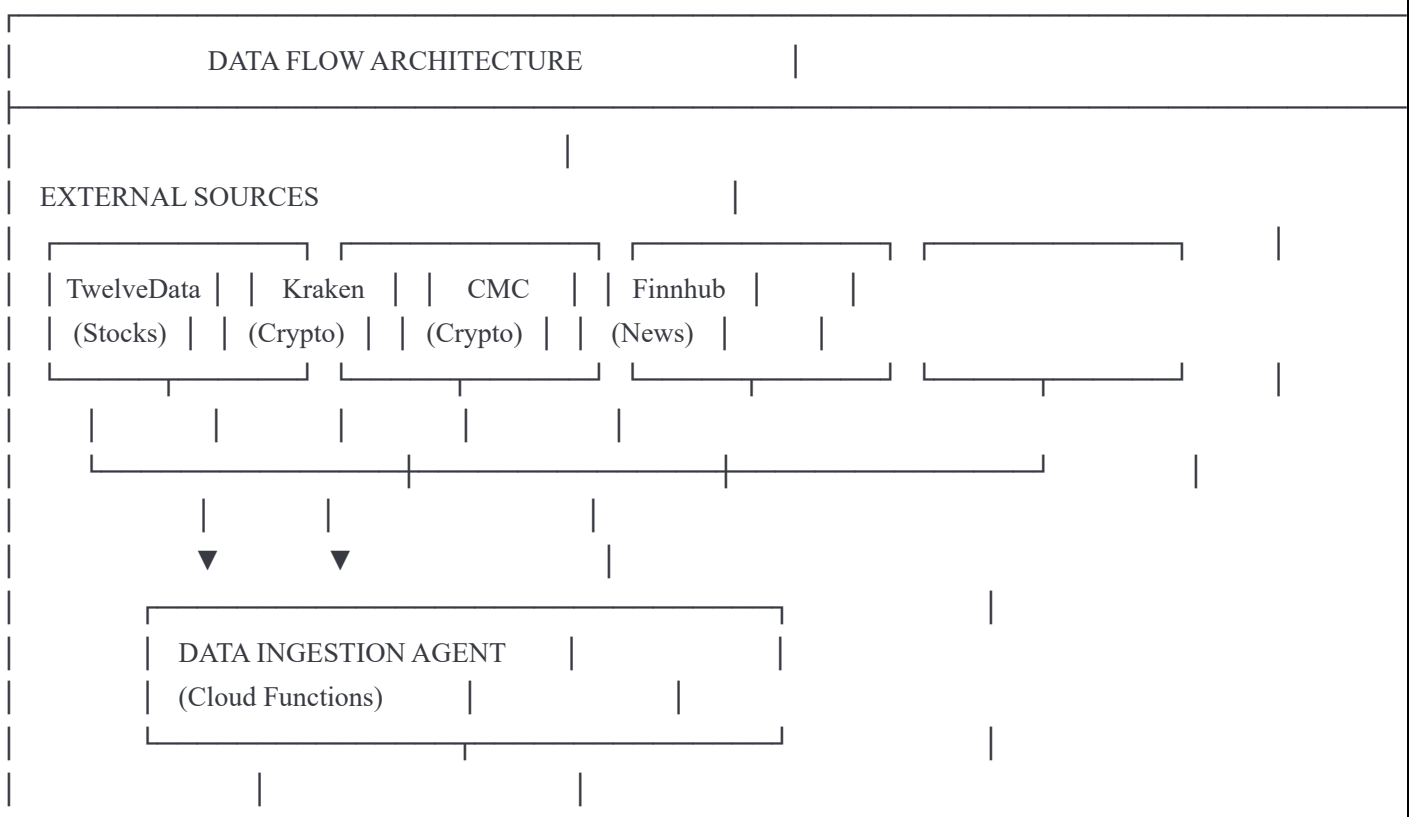
```
-- ============================================
-- CORE OHLCV TABLE (Silver Layer)
-- ============================================
CREATE TABLE aialgotradehits.silver_layer.market_data_daily (
    symbol              STRING NOT NULL,
    asset_class         STRING NOT NULL,  -- 'stock', 'crypto', 'etf', 'forex', 'index', 'commodity'
    datetime            TIMESTAMP NOT NULL,
    open                FLOAT64 NOT NULL,
    high                FLOAT64 NOT NULL,
    low                 FLOAT64 NOT NULL,
    close               FLOAT64 NOT NULL,
    volume              FLOAT64,
    adjusted_close      FLOAT64,
    data_source         STRING,
    ingestion_timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP(),
    quality_score       FLOAT64,

    -- Partitioning for performance
    PRIMARY KEY (symbol, datetime) NOT ENFORCED
)
PARTITION BY DATE(datetime)
CLUSTER BY symbol, asset_class;

-- ============================================
-- FEATURES TABLE (Gold Layer)
-- ============================================
CREATE TABLE aialgotradehits.gold_layer.daily_features_24 (
    symbol              STRING NOT NULL,
    datetime            TIMESTAMP NOT NULL,

    -- Price Data
    open                FLOAT64,
    high                FLOAT64,
    low                 FLOAT64,
    close               FLOAT64,
    volume              FLOAT64,

    -- Momentum Indicators
    rsi_14              FLOAT64,
    stochastic_k        FLOAT64,
    stochastic_d        FLOAT64,
    cci_20              FLOAT64,
    williams_r          FLOAT64,
```

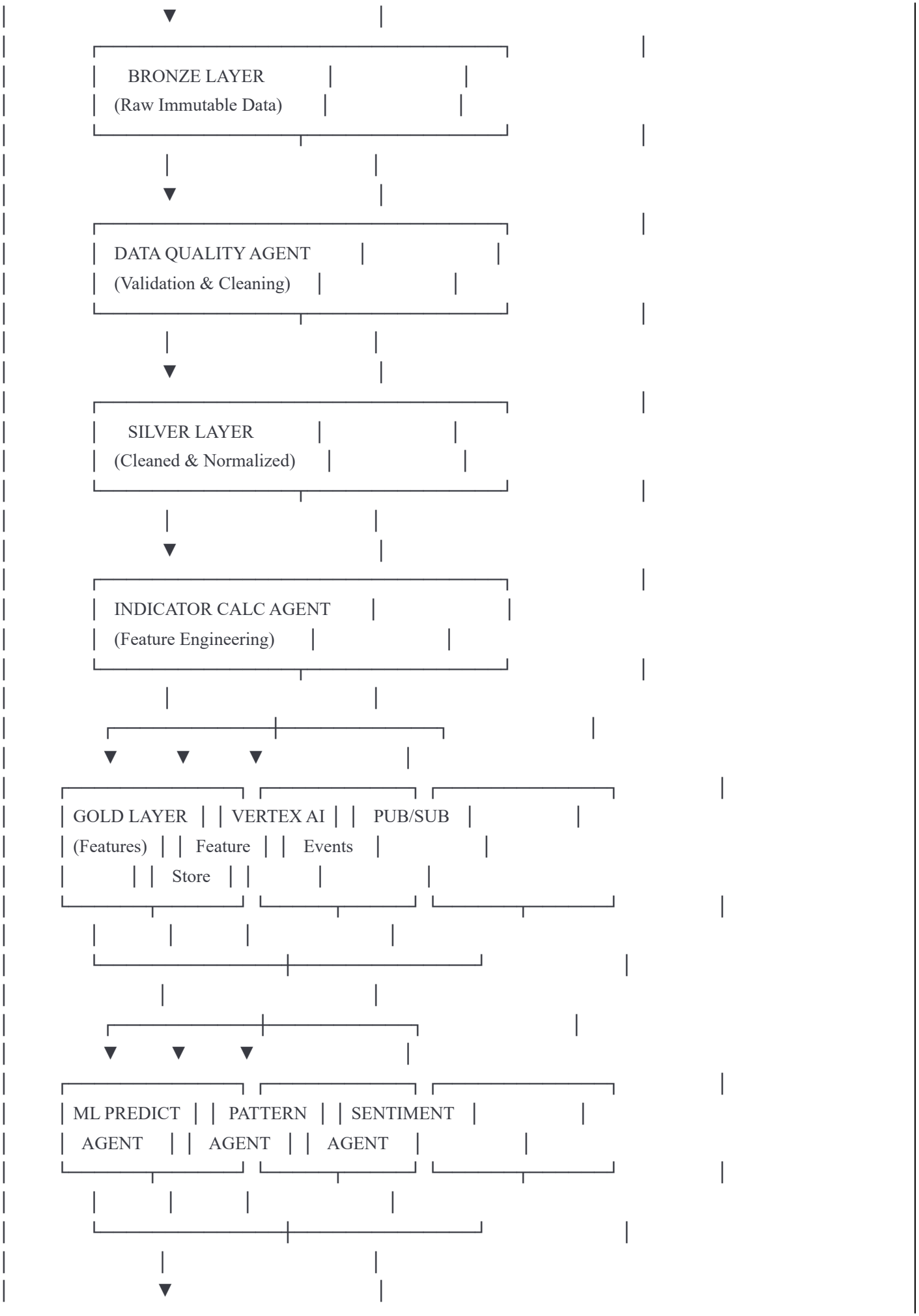
```
    roc_10              FLOAT64,

    -- Trend Indicators
    ema_9               FLOAT64,
    ema_21              FLOAT64,
    ema_50              FLOAT64,
    ema_200             FLOAT64,
    sma_20              FLOAT64,
    sma_50              FLOAT64,
    sma_200             FLOAT64,
    macd_line           FLOAT64,
    macd_signal         FLOAT64,
    macd_histogram      FLOAT64,
    adx_14              FLOAT64,
    psar                FLOAT64,

    -- Volatility Indicators
    atr_14              FLOAT64,
    bollinger_upper     FLOAT64,
    bollinger_middle    FLOAT64,
    bollinger_lower     FLOAT64,
    keltner_upper       FLOAT64,
    keltner_lower       FLOAT64,
    donchian_upper      FLOAT64,
    donchian_lower      FLOAT64,

    -- Volume Indicators
    obv                 FLOAT64,
    vwap                FLOAT64,
    ad_line             FLOAT64,
    cmf                 FLOAT64,
    mfi                 FLOAT64,
    volume_ratio        FLOAT64,

    -- Derived Features
    growth_score        FLOAT64,  -- 0-100
    trend_regime        STRING,   -- 'STRONG_UP', 'UP', 'NEUTRAL', 'DOWN', 'STRONG_DOWN'
    ema_cross_signal    STRING,   -- 'BULLISH_CROSS', 'BEARISH_CROSS', 'NONE'
    golden_cross_flag   BOOLEAN,
    death_cross_flag    BOOLEAN,

    -- Metadata
    calculation_timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP()
)
```

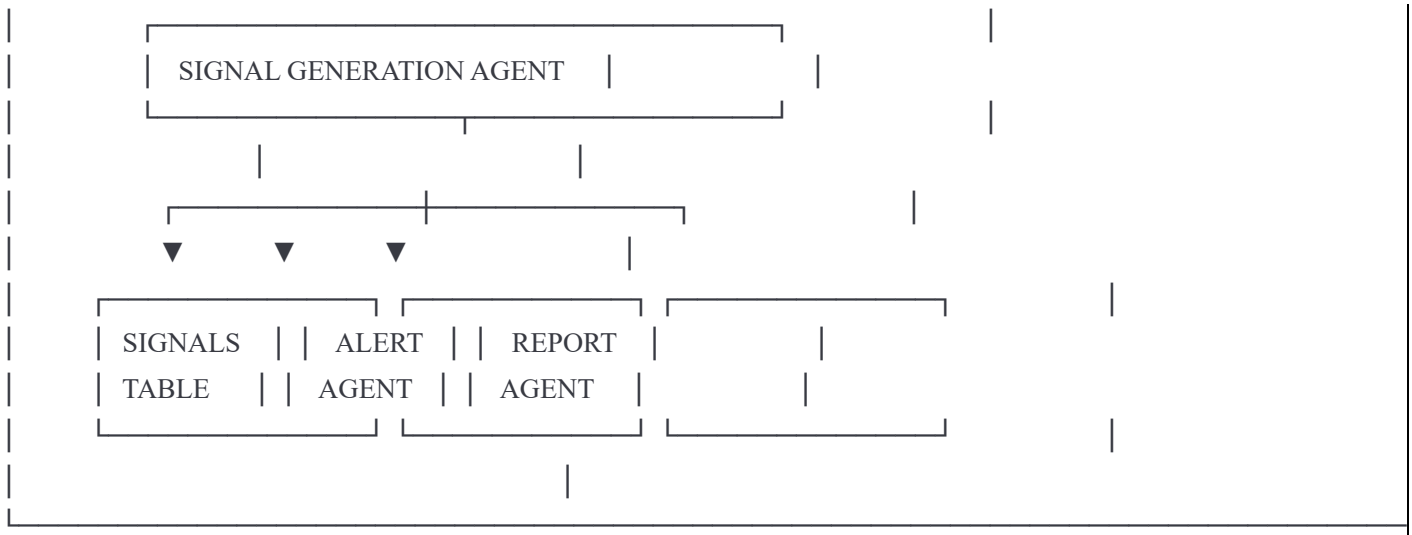
```
PARTITION BY DATE(datetime)
CLUSTER BY symbol;

-- ================================================
-- ML PREDICTIONS TABLE
-- ================================================
CREATE TABLE aialgotradehits.ml_models.predictions (
    prediction_id       STRING NOT NULL,
    symbol              STRING NOT NULL,
    prediction_date     DATE NOT NULL,
    prediction_timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP(),

    -- Predictions
    direction_1d        STRING,     -- 'UP', 'DOWN'
    direction_5d        STRING,
    confidence_1d       FLOAT64,
    confidence_5d       FLOAT64,
    predicted_return_1d FLOAT64,
    predicted_return_5d FLOAT64,

    -- Model Info
    model_version       STRING,
    model_type          STRING,     -- 'xgboost', 'ensemble', 'lstm'
    feature_set         STRING,     -- 'phase1', 'phase1.5', 'full'

    -- Validation (populated after actual results known)
    actual_direction_1d STRING,
    actual_return_1d    FLOAT64,
    prediction_correct  BOOLEAN
)
PARTITION BY prediction_date;

-- ================================================
-- TRADING SIGNALS TABLE
-- ================================================
CREATE TABLE aialgotradehits.trading_data.signals (
    signal_id           STRING NOT NULL,
    symbol              STRING NOT NULL,
    signal_timestamp    TIMESTAMP NOT NULL,

    -- Signal Details
    signal_type         STRING,     -- 'BUY', 'SELL', 'HOLD'
    signal_strength     FLOAT64,    -- 0-100
    signal_source       STRING,     -- 'ml_model', 'pattern', 'sentiment', 'composite'
```

```
    -- Contributing Factors
    ml_signal          STRING,
    ml_confidence      FLOAT64,
    pattern_signal     STRING,
    pattern_name       STRING,
    sentiment_signal   STRING,
    sentiment_score    FLOAT64,

    -- Risk Assessment
    risk_score         FLOAT64,
    suggested_stop_loss FLOAT64,
    suggested_take_profit FLOAT64,

    -- Metadata
    timeframe          STRING,    -- 'daily', 'hourly', '5min'
    is_active          BOOLEAN DEFAULT TRUE,
    expiry_timestamp   TIMESTAMP
)
PARTITION BY DATE(signal_timestamp)
CLUSTER BY symbol;
```

7.3 Data Flow Diagram




BRONZE LAYER
(Raw Immutable Data)
DATA QUALITY AGENT
(Validation & Cleaning)
SILVER LAYER
(Cleaned & Normalized)
INDICATOR CALC AGENT
(Feature Engineering)
GOLD LAYER
(Features)
VERTEX AI
Feature
Store
PUB/SUB
Events
ML PREDICT
AGENT
PATTERN
AGENT
SENTIMENT
AGENT



---

8. AI MODEL REGISTRY & DEPLOYMENT

8.1 Model Registry

yaml

```yaml
model_registry:
  namespace: aialgotradehits
  storage: gs://aialgotradehits-models/

  models:
    direction_predictor_xgboost_v1:
      name: XGBoost Direction Predictor
      version: 1.0.0
      type: classification
      framework: xgboost
      status: PRODUCTION
      metrics:
        accuracy: 0.528
        precision: 0.536
        recall: 0.848
        f1_score: 0.657
      features: 24
      artifact_uri: gs://aialgotradehits-models/xgboost/v1/
      deployed_endpoint: prediction-endpoint-001
      last_trained: "2025-12-13"
      training_data_rows: 53867

    ensemble_predictor_v1:
      name: Ensemble Weighted Predictor
      version: 1.0.0
      type: classification
      framework: custom
      status: STAGING
      components:
        - xgboost: 0.5
        - random_forest: 0.3
        - lstm: 0.2
      target_metrics:
        accuracy: 0.68-0.73
      artifact_uri: gs://aialgotradehits-models/ensemble/v1/

    gemini_market_analyzer:
      name: Gemini Market Analyzer
      version: 2.5-pro
      type: llm
      framework: vertex_ai
      status: PRODUCTION
      model_id: gemini-2.5-pro
```

```
  use_cases:
    - nl2sql
    - market_commentary
    - sentiment_analysis
  temperature: 0.1
  max_tokens: 8192
```

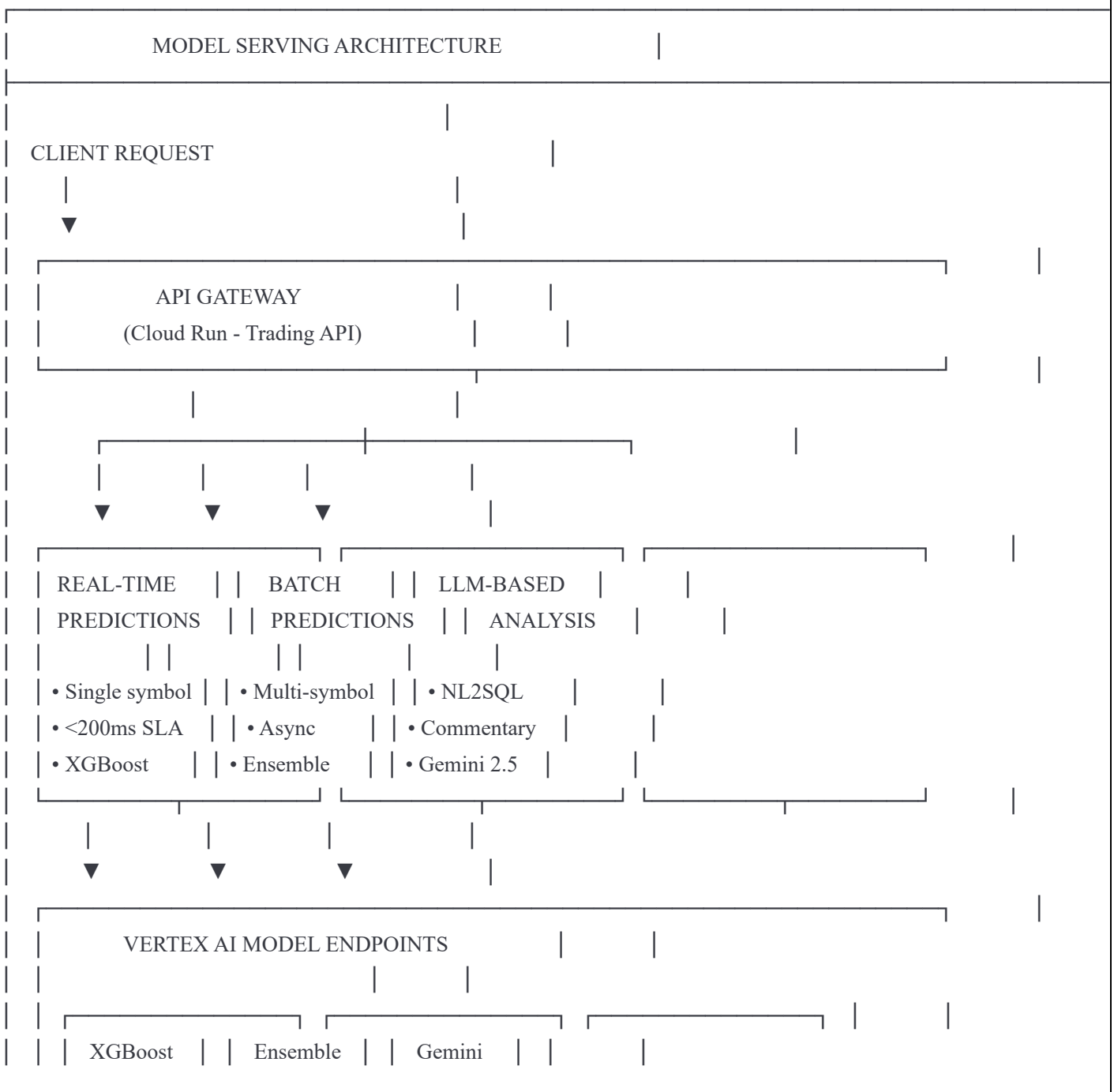
8.2 Model Training Pipeline

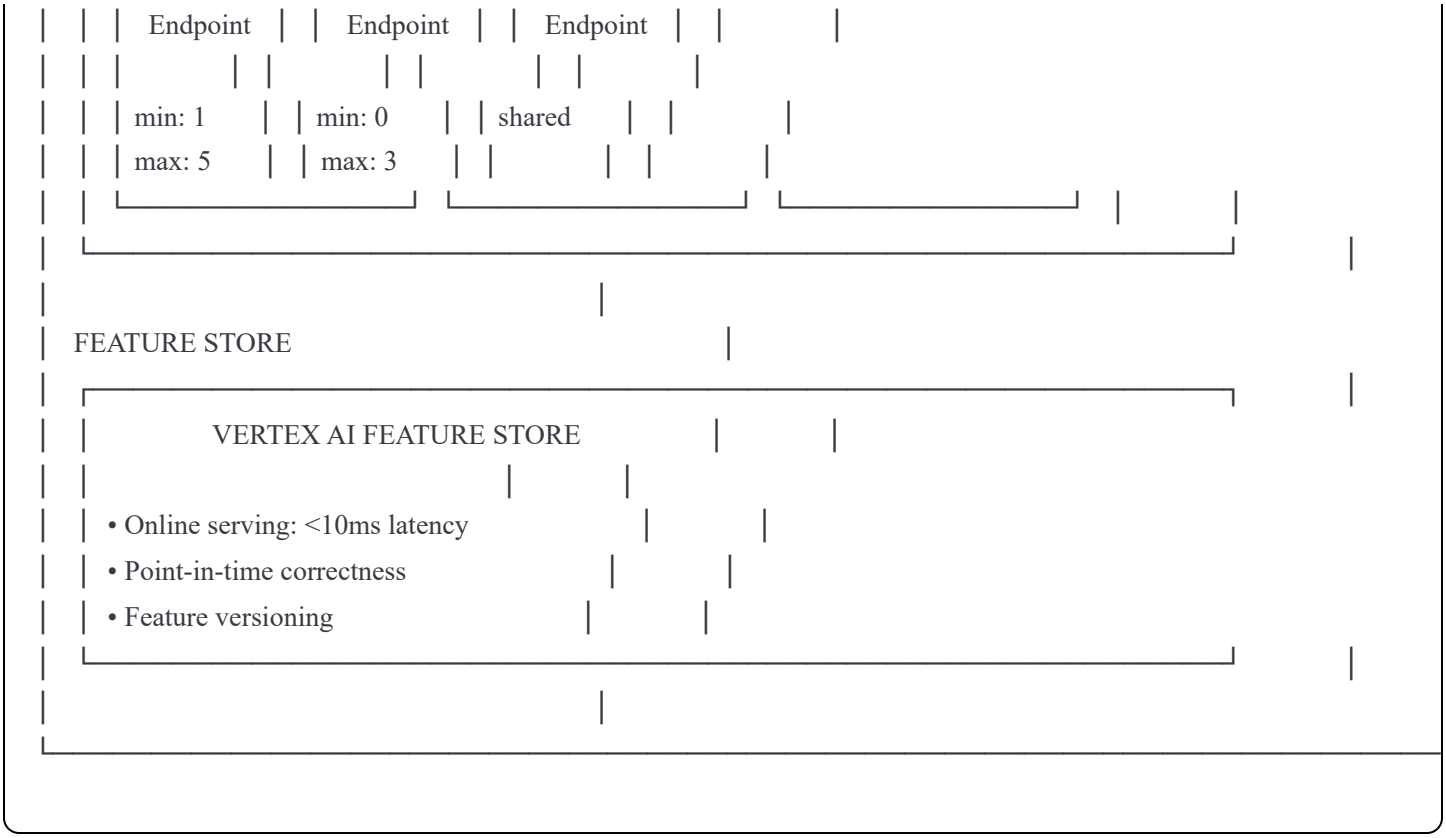
python

```python
# Model Training Agent Implementation
class MLTrainingAgent:
    def __init__(self):
        self.project_id = "cryptobot-462709"
        self.location = "us-central1"
        self.feature_table = "aialgotradehits.gold_layer.daily_features_24"

    async def train_xgboost(self, config: TrainingConfig):
        """Train XGBoost model with specified configuration"""

        # Step 1: Load training data from BigQuery
        query = f"""
        SELECT * FROM `{self.feature_table}`
        WHERE datetime BETWEEN '{config.train_start}' AND '{config.train_end}'
        ORDER BY symbol, datetime
        """
        df = await self.load_from_bigquery(query)

        # Step 2: Feature engineering
        features = self.prepare_features(df, config.feature_set)
        X_train, X_test, y_train, y_test = self.time_based_split(features)

        # Step 3: Train model
        model = xgb.XGBClassifier(
            max_depth=config.max_depth,
            learning_rate=config.learning_rate,
            n_estimators=config.n_estimators,
            min_child_weight=config.min_child_weight,
            subsample=config.subsample,
            colsample_bytree=config.colsample_bytree,
            objective='binary:logistic',
            eval_metric='logloss',
            random_state=42,
            early_stopping_rounds=10
        )

        model.fit(
            X_train, y_train,
            eval_set=[(X_test, y_test)],
            verbose=True
        )

        # Step 4: Evaluate
```

```
        metrics = self.evaluate_model(model, X_test, y_test)

        # Step 5: Save to registry
        artifact_uri = await self.save_model(model, config.version)

        # Step 6: Register in Vertex AI
        await self.register_model(artifact_uri, metrics, config)

        return metrics
```

8.3 Model Serving Architecture





---

9. COMMUNICATION PROTOCOLS & EVENT BUS

9.1 Event Schema Standard

json

```
{
 "$schema": "https://json-schema.org/draft/2020-12/schema",
 "title": "AgentEvent",
 "type": "object",
 "required": ["event_id", "event_type", "timestamp", "source", "payload"],
 "properties": {
  "event_id": {
   "type": "string",
   "format": "uuid"
  },
  "event_type": {
   "type": "string",
   "enum": [
    "CANDLE_INGESTED",
    "INDICATORS_CALCULATED",
    "PATTERN_DETECTED",
    "PREDICTION_GENERATED",
    "SIGNAL_CREATED",
    "ALERT_TRIGGERED",
    "MODEL_UPDATED",
    "AGENT_HEALTH_CHECK",
    "WORKFLOW_STARTED",
    "WORKFLOW_COMPLETED",
    "ERROR_OCCURRED"
   ]
  },
  "timestamp": {
   "type": "string",
   "format": "date-time"
  },
  "source": {
   "type": "object",
   "properties": {
    "agent_id": {"type": "string"},
    "agent_type": {"type": "string"},
    "version": {"type": "string"}
   }
  },
  "payload": {
   "type": "object"
  },
  "metadata": {
   "type": "object",
```

```
    "properties": {
      "correlation_id": {"type": "string"},
      "trace_id": {"type": "string"},
      "priority": {"type": "string", "enum": ["LOW", "NORMAL", "HIGH", "CRITICAL"]},
      "ttl_seconds": {"type": "integer"}
    }
  }
 }
}
```

9.2 Event Examples

json

```
// CANDLE_INGESTED Event
{
  "event_id": "550e8400-e29b-41d4-a716-446655440001",
  "event_type": "CANDLE_INGESTED",
  "timestamp": "2026-01-03T00:05:00Z",
  "source": {
    "agent_id": "data-ingestion-001",
    "agent_type": "DATA_INGESTION",
    "version": "1.0.0"
  },
  "payload": {
    "symbol": "BTCUSD",
    "asset_class": "crypto",
    "timeframe": "daily",
    "candle": {
      "datetime": "2026-01-02T00:00:00Z",
      "open": 95000.0,
      "high": 96500.0,
      "low": 94200.0,
      "close": 95800.0,
      "volume": 25000000000
    },
    "data_source": "kraken",
    "quality_score": 0.98
  },
  "metadata": {
    "correlation_id": "daily-collection-20260103",
    "trace_id": "trace-abc123",
    "priority": "HIGH"
  }
}

// SIGNAL_CREATED Event
{
  "event_id": "550e8400-e29b-41d4-a716-446655440002",
  "event_type": "SIGNAL_CREATED",
  "timestamp": "2026-01-03T00:15:00Z",
  "source": {
    "agent_id": "signal-generator-001",
    "agent_type": "SIGNAL_GENERATION",
    "version": "1.0.0"
  },
  "payload": {
```

```
    "signal_id": "sig-12345",
    "symbol": "BTCUSD",
    "signal_type": "BUY",
    "signal_strength": 82.5,
    "components": {
      "ml_prediction": {"direction": "UP", "confidence": 0.75},
      "pattern_signal": {"type": "BULLISH_ENGULFING", "confidence": 0.85},
      "sentiment_signal": {"score": 0.65, "trend": "IMPROVING"}
    },
    "risk_assessment": {
      "risk_score": 35,
      "suggested_stop_loss": 93500,
      "suggested_take_profit": 100000
    },
    "expiry": "2026-01-04T00:00:00Z"
  },
  "metadata": {
    "correlation_id": "daily-collection-20260103",
    "trace_id": "trace-abc123",
    "priority": "CRITICAL"
  }
}
```


9.3 Inter-Agent Communication Patterns

python

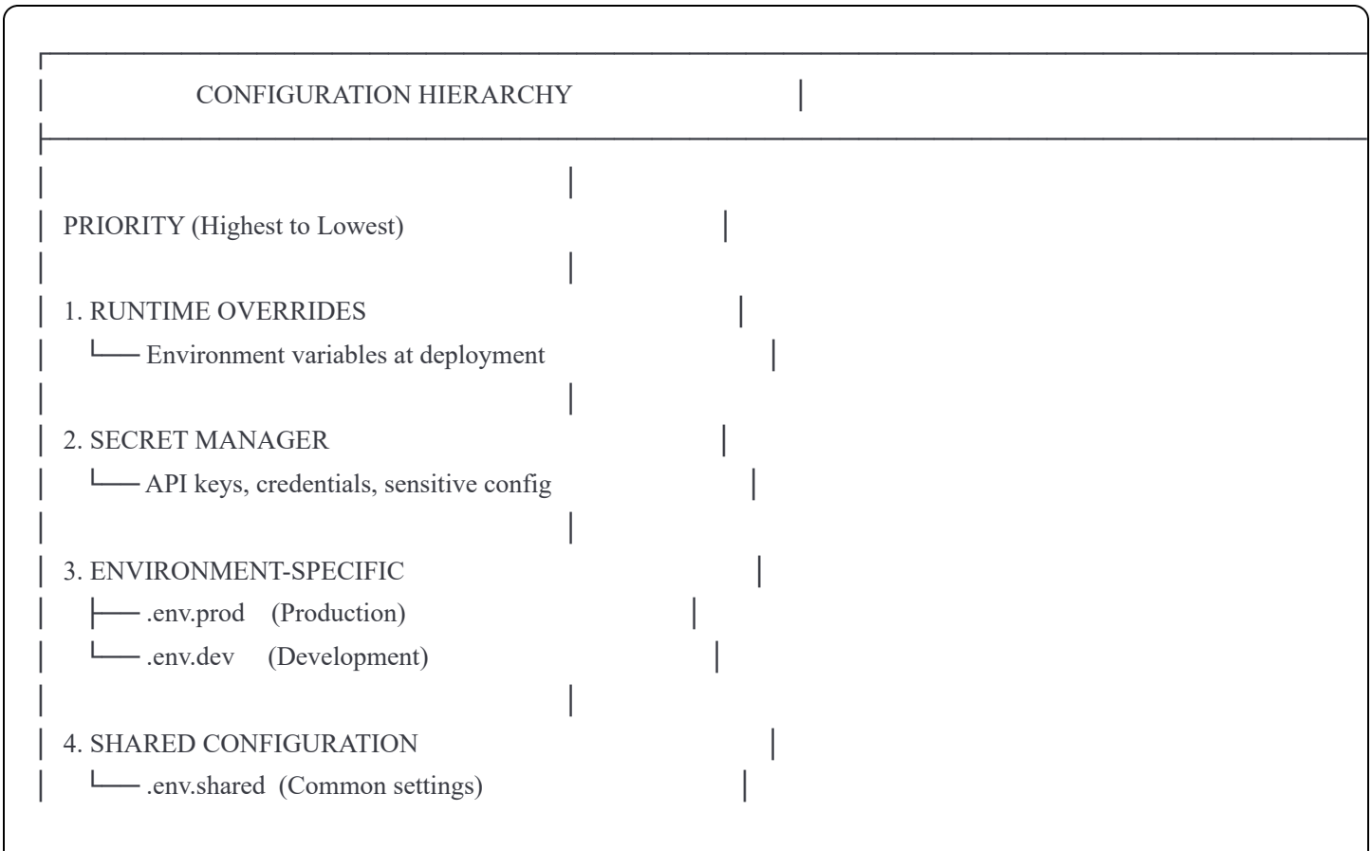
```python
# Pub/Sub Event Publisher
class EventPublisher:
    def __init__(self, project_id: str):
        self.publisher = pubsub_v1.PublisherClient()
        self.project_id = project_id

    async def publish(self, topic: str, event: AgentEvent):
        topic_path = self.publisher.topic_path(self.project_id, topic)

        # Serialize event
        data = json.dumps(event.to_dict()).encode('utf-8')

        # Add attributes for filtering
        attributes = {
            'event_type': event.event_type,
            'source_agent': event.source.agent_id,
            'priority': event.metadata.get('priority', 'NORMAL')
        }

        # Publish with retry
        future = self.publisher.publish(
            topic_path,
            data,
            **attributes
        )

        return await asyncio.wrap_future(future)


# Pub/Sub Event Subscriber
class EventSubscriber:
    def __init__(self, project_id: str, subscription_id: str):
        self.subscriber = pubsub_v1.SubscriberClient()
        self.subscription_path = self.subscriber.subscription_path(
            project_id, subscription_id
        )
        self.handlers = {}

    def register_handler(self, event_type: str, handler: Callable):
        self.handlers[event_type] = handler

    def callback(self, message):
        try:
```

```
            event = AgentEvent.from_json(message.data.decode('utf-8'))

            handler = self.handlers.get(event.event_type)
            if handler:
                asyncio.run(handler(event))

            message.ack()
        except Exception as e:
            logging.error(f"Error processing message: {e}")
            message.nack()

    def start_listening(self):
        streaming_pull_future = self.subscriber.subscribe(
            self.subscription_path,
            callback=self.callback
        )
        return streaming_pull_future
```

---

10. CONFIGURATION MANAGEMENT

10.1 Configuration Hierarchy



```
│                                                                              │
│  5. DEFAULT VALUES                                                           │
│     └── config.yaml  (Application defaults)                                  │
│                                                                              │
└──────────────────────────────────────────────────────────────────────────────┘
```

10.2 Master Configuration File

yaml

```yaml
# ============================================
# AIALGOTRADEHITS MASTER CONFIGURATION
# Version: 6.0
# ============================================

app:
  name: AIAlgoTradeHits
  version: "6.0.0"
  environment: ${ENVIRONMENT}  # 'development' or 'production'

# ============================================
# GCP INFRASTRUCTURE
# ============================================
gcp:
  project_id: cryptobot-462709
  region: us-central1

  cloud_run:
    services:
      orchestrator:
        name: orchestrator-agent
        url: ${ORCHESTRATOR_URL}
        min_instances: 1
        max_instances: 10
      trading_api:
        name: trading-api
        url: https://trading-api-1075463475276.us-central1.run.app
        min_instances: 1
        max_instances: 20

  bigquery:
    datasets:
      trading: aialgotradehits.trading_data
      ml_models: aialgotradehits.ml_models
      bronze: aialgotradehits.bronze_layer
      silver: aialgotradehits.silver_layer
      gold: aialgotradehits.gold_layer

  pubsub:
    topics:
      - candle-ingested
      - indicators-calculated
      - pattern-detected
```

```yaml
    - prediction-generated
    - signal-generated
    - agent-events

  vertex_ai:
    endpoints:
      prediction: ${VERTEX_PREDICTION_ENDPOINT}
    models:
      gemini: gemini-2.5-pro

# ================================================
# EXTERNAL APIS
# ================================================
apis:
  twelvedata:
    base_url: https://api.twelvedata.com
    api_key: ${TWELVEDATA_API_KEY}
    plan: Pro
    daily_limit: 1152000
    rate_limit_per_min: 800
    batch_size: 8

  kraken:
    ws_url: wss://ws.kraken.com
    rest_url: https://api.kraken.com
    api_key: ${KRAKEN_API_KEY}
    api_secret: ${KRAKEN_API_SECRET}

  coinmarketcap:
    base_url: https://pro-api.coinmarketcap.com
    api_key: ${CMC_API_KEY}
    plan: Basic
    daily_limit: 333

  finnhub:
    base_url: https://finnhub.io/api/v1
    api_key: ${FINNHUB_API_KEY}
    rate_limit_per_min: 60

# ================================================
# AGENT CONFIGURATION
# ================================================
agents:
  orchestrator:
```

```yaml
  id: orchestrator-001
  health_check_interval: 30s
  circuit_breaker:
    failure_threshold: 5
    recovery_timeout: 60s

 data_ingestion:
  id: data-ingestion-001
  parallel_workers: 20
  retry_attempts: 3
  retry_backoff: exponential

 indicator_calc:
  id: indicator-calc-001
  indicator_sets:
    daily: 24
    hourly: 12
    5min: 8
    1min: 5

 ml_prediction:
  id: ml-prediction-001
  default_model: xgboost
  confidence_threshold: 0.6
  batch_size: 100

 nl2sql:
  id: nl2sql-001
  model: gemini-2.5-pro
  temperature: 0.1
  max_tokens: 8192
  max_result_rows: 10000

# ============================================
# ML MODELS
# ============================================
ml:
 training:
  schedule: "0 2 * * 0"  # Weekly Sunday 2 AM
  min_training_samples: 5000
  validation_split: 0.2
  test_split: 0.1

 xgboost:
```

```yaml
    max_depth: 5
    learning_rate: 0.1
    n_estimators: 100
    min_child_weight: 50
    subsample: 0.8
    colsample_bytree: 0.8

  targets:
    accuracy: 0.66  # 66%
    precision: 0.60
    recall: 0.70

# ================================================
# MONITORING & ALERTING
# ================================================
monitoring:
  health_check:
    interval: 30s
    timeout: 10s

  metrics:
    collection_interval: 60s
    retention_days: 90

  alerting:
    channels:
      - email: admin@aialgotradehits.com
      - slack: ${SLACK_WEBHOOK}
    thresholds:
      error_rate: 0.01  # 1%
      latency_p99: 2000  # 2 seconds
      api_quota_warning: 0.8  # 80%

# ================================================
# FEATURE FLAGS
# ================================================
features:
  enable_realtime_streaming: true
  enable_sentiment_analysis: true
  enable_pattern_detection: true
  enable_auto_trading: false  # Safety: manual approval required
  enable_advanced_models: true
```

10.3 Secret Management

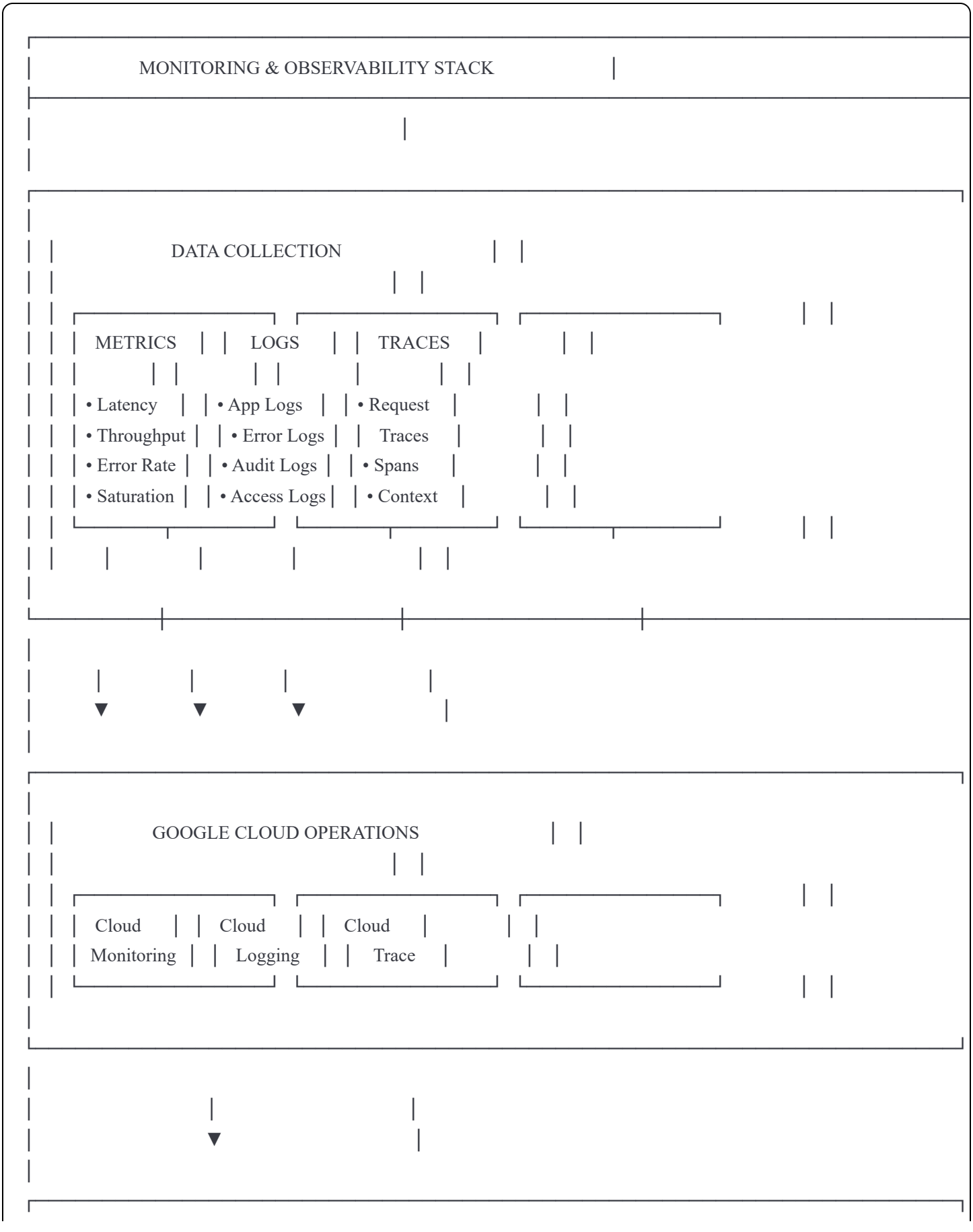
yaml

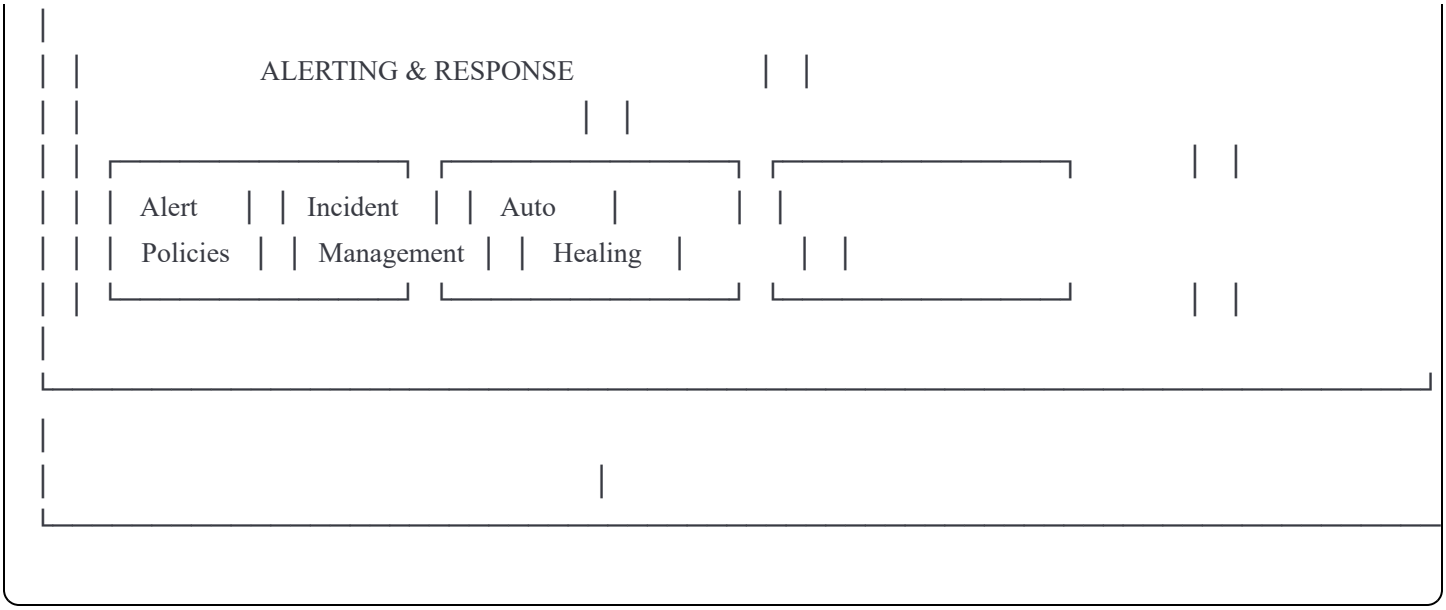
```yaml
# GCP Secret Manager Configuration
secrets:
  # API Keys
  TWELVEDATA_API_KEY:
    version: latest
    path: projects/cryptobot-462709/secrets/TWELVEDATA_API_KEY

  KRAKEN_API_KEY:
    version: latest
    path: projects/cryptobot-462709/secrets/KRAKEN_API_KEY

  KRAKEN_API_SECRET:
    version: latest
    path: projects/cryptobot-462709/secrets/KRAKEN_API_SECRET

  CMC_API_KEY:
    version: latest
    path: projects/cryptobot-462709/secrets/CMC_API_KEY

  FINNHUB_API_KEY:
    version: latest
    path: projects/cryptobot-462709/secrets/FINNHUB_API_KEY

  # Cloud Service Keys
  VERTEX_AI_SERVICE_ACCOUNT:
    version: latest
    path: projects/cryptobot-462709/secrets/VERTEX_AI_SA_KEY

  # Notification Keys
  SLACK_WEBHOOK:
    version: latest
    path: projects/cryptobot-462709/secrets/SLACK_WEBHOOK

  SENDGRID_API_KEY:
    version: latest
    path: projects/cryptobot-462709/secrets/SENDGRID_API_KEY
```

---

11. MONITORING, OBSERVABILITY & ALERTING

11.1 Monitoring Architecture





11.2 Key Metrics Dashboard

yaml

```yaml
dashboards:
  agent_health:
    name: Agent Health Overview
    widgets:
      - title: Agent Availability
        type: uptime
        targets:
          - orchestrator-001
          - data-ingestion-001
          - ml-prediction-001
        threshold: 99.9%

      - title: Agent Response Times
        type: latency_heatmap
        percentiles: [p50, p90, p99]
        threshold_p99: 2000ms

      - title: Error Rates by Agent
        type: time_series
        metric: error_count
        group_by: agent_id
        threshold: 1%

  data_pipeline:
    name: Data Pipeline Health
    widgets:
      - title: Records Ingested
        type: counter
        metric: candles_ingested
        period: 24h

      - title: API Quota Usage
        type: gauge
        metrics:
          - twelvedata_quota_used
          - twelvedata_quota_limit
        warning: 80%
        critical: 95%

      - title: Data Quality Score
        type: gauge
        metric: avg_quality_score
        threshold: 95%
```

```yaml
ml_performance:
  name: ML Model Performance
  widgets:
    - title: Prediction Accuracy
      type: gauge
      metric: model_accuracy
      target: 66%
      current: 52.8%

    - title: Predictions Generated
      type: counter
      metric: predictions_count
      period: 24h

    - title: Model Inference Latency
      type: histogram
      metric: inference_latency_ms
      buckets: [50, 100, 200, 500, 1000]
```

11.3 Alert Policies

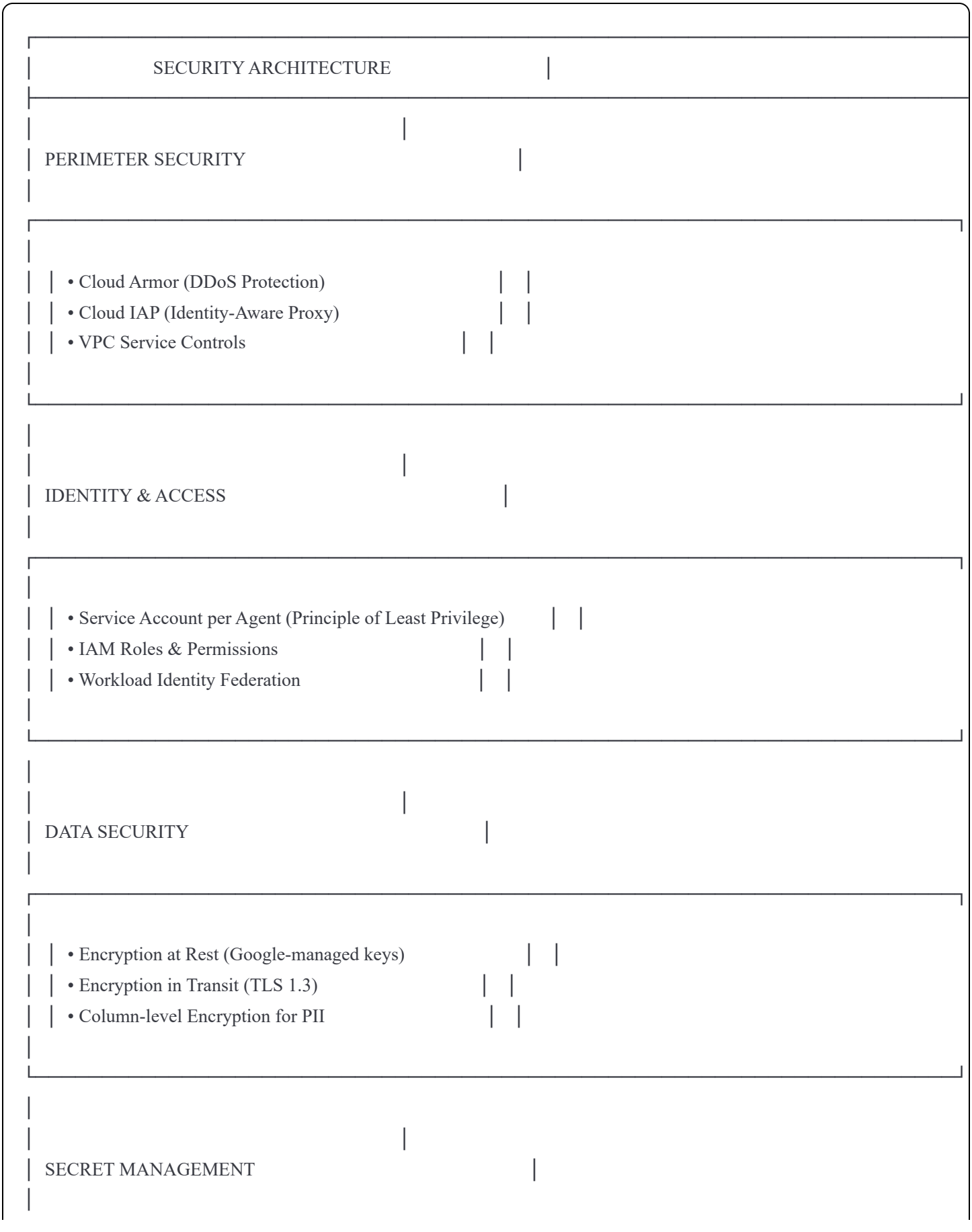
yaml

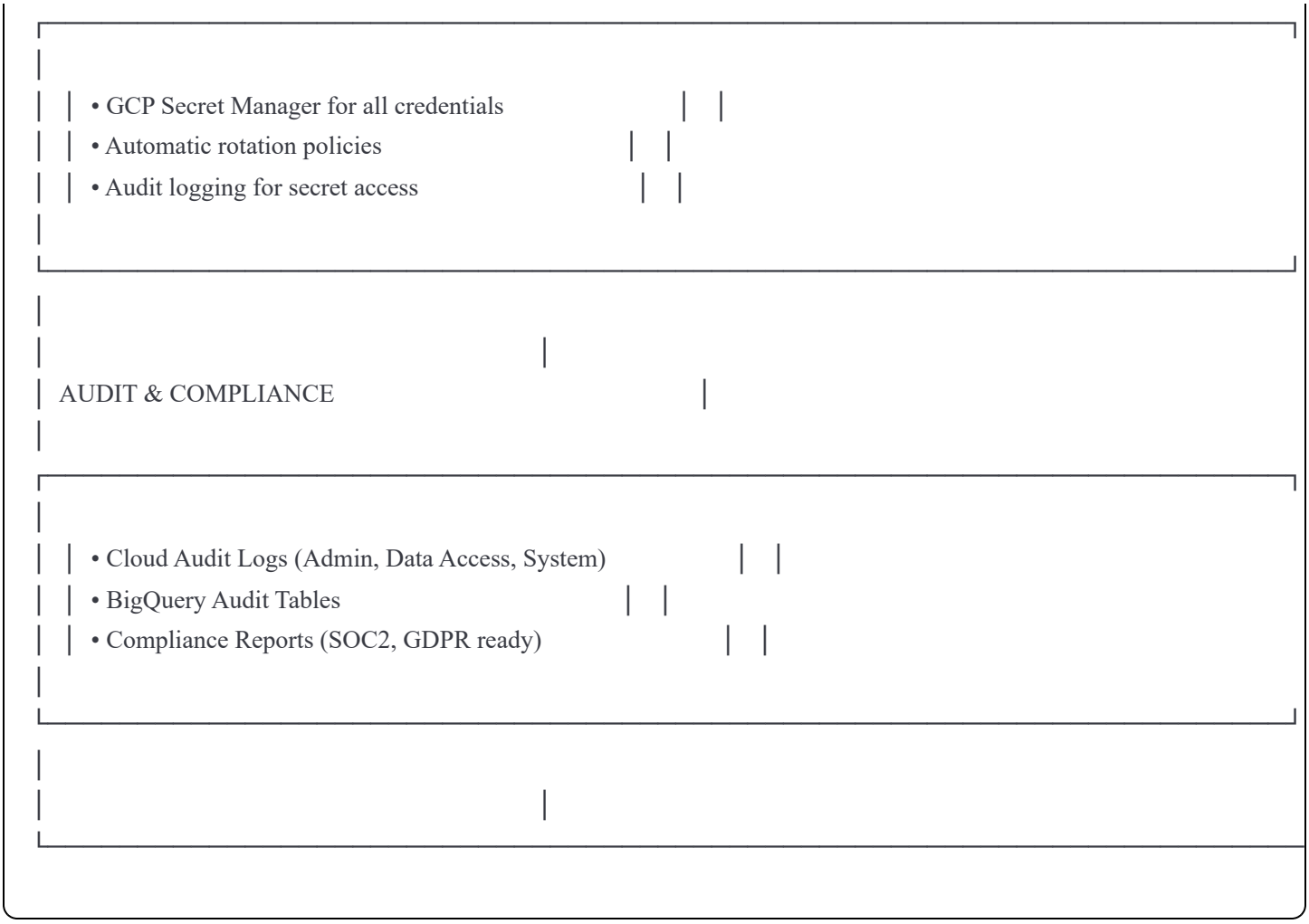
```yaml
alert_policies:
  critical:
    agent_down:
      condition: uptime < 95%
      duration: 5m
      channels: [pagerduty, slack, email]
      severity: P1

    high_error_rate:
      condition: error_rate > 5%
      duration: 5m
      channels: [pagerduty, slack]
      severity: P1

    api_quota_exhausted:
      condition: api_quota_remaining < 5%
      channels: [slack, email]
      severity: P1

  warning:
    degraded_performance:
      condition: latency_p99 > 2000ms
      duration: 10m
      channels: [slack]
      severity: P2

    model_accuracy_drop:
      condition: accuracy < baseline - 5%
      duration: 1h
      channels: [email]
      severity: P3

    api_quota_warning:
      condition: api_quota_remaining < 20%
      channels: [slack]
      severity: P3
```

---

12. SECURITY & COMPLIANCE FRAMEWORK

12.1 Security Architecture





12.2 Service Account Configuration

yaml

```
service_accounts:
  orchestrator-agent-sa:
    email: orchestrator@cryptobot-462709.iam.gserviceaccount.com
    roles:
      - roles/pubsub.publisher
      - roles/pubsub.subscriber
      - roles/bigquery.dataViewer
      - roles/secretmanager.secretAccessor
      - roles/run.invoker

  data-ingestion-sa:
    email: data-ingestion@cryptobot-462709.iam.gserviceaccount.com
    roles:
      - roles/bigquery.dataEditor
      - roles/pubsub.publisher
      - roles/secretmanager.secretAccessor

  ml-prediction-sa:
    email: ml-prediction@cryptobot-462709.iam.gserviceaccount.com
    roles:
      - roles/bigquery.dataViewer
      - roles/aiplatform.user
      - roles/pubsub.publisher

  nl2sql-sa:
    email: nl2sql@cryptobot-462709.iam.gserviceaccount.com
    roles:
      - roles/bigquery.jobUser
      - roles/bigquery.dataViewer
      - roles/aiplatform.user
```

---

13. COST OPTIMIZATION STRATEGIES

13.1 Current Cost Breakdown

```
┌──────────────────────────────────────────────────────────────────────────────┐
│                    MONTHLY COST BREAKDOWN                                    │
├──────────────────────────────────────────────────────────────────────────────┤
│                                                                              │
│  SERVICE                CURRENT        OPTIMIZED       SAVINGS             │
```

```
│
├──────────────────────────────────────────────────────────────────────────────┤
│                                                                              │
│                                                                              │
│ Cloud Functions (Gen2)    $126          $100          $26 (21%)              │
│ Cloud Run                 $50-100       $40-60        $30 (35%)              │
│ BigQuery Storage          $2-5          $2-3          $2 (40%)               │
│ BigQuery Queries          $50-100       $30-50        $30 (40%)              │
│ Vertex AI                 $200-500      $150-300      $100 (30%)             │
│ Gemini API                $100-300      $80-200       $50 (25%)              │
│ Cloud Storage             $5            $3            $2 (40%)               │
│ Cloud Scheduler           $0.30         $0.30         $0                     │
│ Pub/Sub                   $10           $8            $2 (20%)               │
│                                                                              │
├──────────────────────────────────────────────────────────────────────────────┤
│                                                                              │
│ TOTAL (Monthly)           $543-1,140    $413-724      ~$200 (25%)            │
│                                                                              │
│ EXTERNAL SUBSCRIPTIONS                                                       │
│                                                                              │
├──────────────────────────────────────────────────────────────────────────────┤
│                                                                              │
│ TwelveData Pro            $229/month                                         │
│                                                                              │
└──────────────────────────────────────────────────────────────────────────────┘
```

13.2 Optimization Strategies

yaml

```yaml
cost_optimization:
  bigquery:
    strategies:
      - Use partitioned tables (by date)
      - Cluster by frequently queried columns
      - Set table expiration for temp data
      - Use BI Engine for dashboards
      - Avoid SELECT * queries

    implementation:
      partition_expiration_days: 365
      clustering_columns: [symbol, asset_class]
      bi_engine_reserved_gb: 1

  cloud_functions:
    strategies:
      - Right-size memory allocation
      - Optimize cold start times
      - Use min instances sparingly
      - Batch operations where possible

    implementation:
      memory_per_function:
        daily_fetcher: 512MB
        hourly_fetcher: 256MB
        indicator_calc: 1GB
        pattern_detect: 512MB

  vertex_ai:
    strategies:
      - Use batch predictions for non-urgent
      - Cache model predictions
      - Optimize model size
      - Use spot instances for training

    implementation:
      batch_prediction_threshold: 100  # symbols
      cache_ttl_minutes: 60
      spot_training: enabled

  api_usage:
    strategies:
      - Maximize TwelveData batch requests (8 symbols)
```

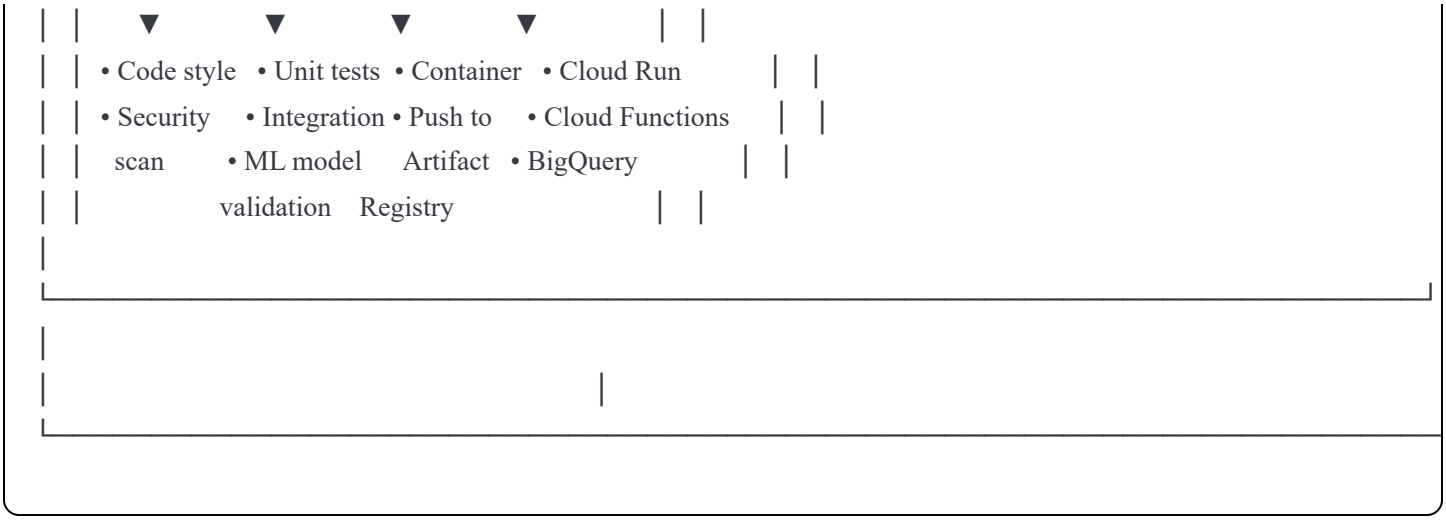
- Implement intelligent caching
- Prioritize high-value symbols
- Use webhooks over polling

current_utilization: 0.63%
target_utilization: 50%
daily_credits_available: 1,152,000

---

14. DEPLOYMENT PIPELINES

14.1 CI/CD Architecture



14.2 Deployment Commands

bash

```
# Deploy all Cloud Functions
./deploy_functions.sh

# Deploy specific agent
gcloud run deploy trading-api \
  --image gcr.io/cryptobot-462709/trading-api:latest \
  --region us-central1 \
  --platform managed \
  --allow-unauthenticated

# Deploy Cloud Scheduler jobs
gcloud scheduler jobs create http daily-crypto-fetch-job \
  --schedule="0 0 * * *" \
  --uri="https://us-central1-cryptobot-462709.cloudfunctions.net/daily-crypto-fetcher" \
  --location=us-central1

# Deploy ML model to Vertex AI
gcloud ai endpoints deploy-model $ENDPOINT_ID \
  --model=$MODEL_ID \
  --display-name="xgboost-predictor" \
  --machine-type=n1-standard-4 \
  --min-replica-count=1 \
  --max-replica-count=5 \
  --traffic-split=0=100

# Update BigQuery tables
bq update --schema schema.json cryptobot-462709:aialgotradehits.daily_features_24
```

15. API REFERENCE & ENDPOINTS

15.1 Trading API Endpoints

yaml

```yaml
base_url: https://trading-api-1075463475276.us-central1.run.app

endpoints:
  # AI Trading Signals
  GET /api/ai/trading-signals:
    description: Generate buy/sell signals
    parameters:
      - symbol: string (optional)
      - timeframe: string (daily|hourly|5min)
      - limit: integer (default: 20)
    response:
      - signal_id
      - symbol
      - signal_type
      - confidence
      - timestamp

  # Rise Cycle Candidates
  GET /api/ai/rise-cycle-candidates:
    description: EMA crossover cycle detection
    parameters:
      - asset_class: string (optional)
      - min_score: integer (default: 50)
    response:
      - candidates[]
        - symbol
        - growth_score
        - ema_cross_type
        - days_since_cross

  # ML Predictions
  GET /api/ai/ml-predictions:
    description: Growth score predictions
    parameters:
      - symbols: string[] (optional)
      - limit: integer (default: 50)
    response:
      - predictions[]
        - symbol
        - direction
        - confidence
        - growth_score
```

```
# Growth Screener
GET /api/ai/growth-screener:
  description: High growth score scanner
  parameters:
    - min_score: integer (default: 70)
    - asset_class: string (optional)
    - sort_by: string (score|confidence)
  response:
    - results[]
      - symbol
      - growth_score
      - trend_regime
      - volume_ratio

# Natural Language Query
POST /api/ai/text-to-sql:
  description: Natural language to SQL
  body:
    query: string
  response:
    - sql: string
    - results[]
    - interpretation: string

# Symbol Analysis
GET /api/ai/analyze-symbol:
  description: AI-powered symbol analysis
  parameters:
    - symbol: string (required)
    - depth: string (basic|detailed)
  response:
    - symbol
    - current_price
    - indicators
    - patterns
    - sentiment
    - recommendation

# Market Summary
GET /api/ai/market-summary:
  description: AI market overview
  parameters:
    - asset_class: string (optional)
  response:
```

- summary
- top_gainers
- top_losers
- sector_performance
- sentiment_overview

15.2 API Usage Examples

bash

```
# Get rise cycle candidates
curl "https://trading-api-1075463475276.us-central1.run.app/api/ai/rise-cycle-candidates"

# Get ML predictions with growth scores
curl "https://trading-api-1075463475276.us-central1.run.app/api/ai/ml-predictions?limit=20"

# Screen for high growth scores
curl "https://trading-api-1075463475276.us-central1.run.app/api/ai/growth-screener?min_score=75"

# Natural language query
curl -X POST "https://trading-api-1075463475276.us-central1.run.app/api/ai/text-to-sql" \
  -H "Content-Type: application/json" \
  -d '{"query": "show me oversold cryptos with high volume"}'

# Analyze specific symbol
curl "https://trading-api-1075463475276.us-central1.run.app/api/ai/analyze-symbol?symbol=BTCUSD&depth=detailed"

# Get market summary
curl "https://trading-api-1075463475276.us-central1.run.app/api/ai/market-summary?asset_class=crypto"
```

---

16. IMPLEMENTATION ROADMAP

16.1 Phase Timeline

```
┌──────────────────────────────────────────────────────────────────────────────┐
│                        IMPLEMENTATION ROADMAP                                │
├──────────────────────────────────────────────────────────────────────────────┤
│                                                                              │
│  Q1 2026 (Jan-Mar)                                                           │
│
```

PHASE 1: FOUNDATION ENHANCEMENT

- [x] Deploy Orchestrator Agent
- [x] Implement Event Bus (Pub/Sub)
- [ ] Upgrade Data Ingestion to 50% API capacity
- [ ] Deploy all Phase 1.5 indicators
- [ ] Improve ML model to 60% accuracy

Q2 2026 (Apr-Jun)

PHASE 2: INTELLIGENCE LAYER

- [ ] Deploy Pattern Recognition Agent
- [ ] Deploy Sentiment Analysis Agent
- [ ] Implement Multi-Agent Coordination
- [ ] Achieve 66% ML accuracy target
- [ ] Launch Real-Time Analytics Engine

Q3 2026 (Jul-Sep)

PHASE 3: ADVANCED FEATURES

- [ ] Deploy Backtesting Simulator
- [ ] Implement Portfolio Management Agent
- [ ] Launch AIAlgoTradeHits.org Training Portal
- [ ] Achieve 72% ML accuracy target
- [ ] Advanced Ensemble Models

Q4 2026 (Oct-Dec)

PHASE 4: EXECUTION & SCALE

- [ ] Paper Trading Integration (Kraken/Alpaca)
- [ ] Advanced Risk Management Agent
- [ ] Treasury Intelligence System (Phase 2)
- [ ] Institutional Features
- [ ] Full Production Deployment

16.2 Success Metrics

```yaml
success_metrics:
  q1_2026:
    ml_accuracy: 60%
    api_utilization: 25%
    agents_deployed: 8
    data_coverage: 1500+ symbols

  q2_2026:
    ml_accuracy: 66%
    api_utilization: 50%
    agents_deployed: 12
    real_time_latency_p99: 500ms

  q3_2026:
    ml_accuracy: 70%
    api_utilization: 75%
    backtest_win_rate: 60%
    user_satisfaction: 4.5/5

  q4_2026:
    ml_accuracy: 72%
    api_utilization: 90%
    paper_trading_pnl: positive
    system_uptime: 99.9%
```

APPENDIX

A. Glossary of Terms

Term	Definition
Agent	Autonomous software component with specific responsibilities
Orchestrator	Master agent coordinating all other agents
Feature Store	Centralized repository for ML features

Term	Definition
Growth Score	Composite indicator (0-100) measuring trend strength
Pub/Sub	Google Cloud message queue service
Bronze/Silver/Gold	Data quality layers in the architecture
NL2SQL	Natural Language to SQL query conversion

B. Contact & Support

- **Technical Lead**: Irfan
- **Platform**: AIAlgoTradeHits.com
- **GCP Project**: cryptobot-462709
- **Documentation**: /Trading/masterquery.md

C. Version History

Version	Date	Changes
6.0	Jan 2026	Full Agentic AI Architecture
5.0	Dec 2025	Trading Execution Integration
4.0	Dec 2025	ML Model Training Complete
3.0	Nov 2025	Configuration Management
2.0	Oct 2025	Multi-Asset Support
1.0	Sep 2025	Initial Release

Platform: AIAlgoTradeHits.com
Architecture: Agentic AI
Last Updated: January 3, 2026
Document Version: 6.0