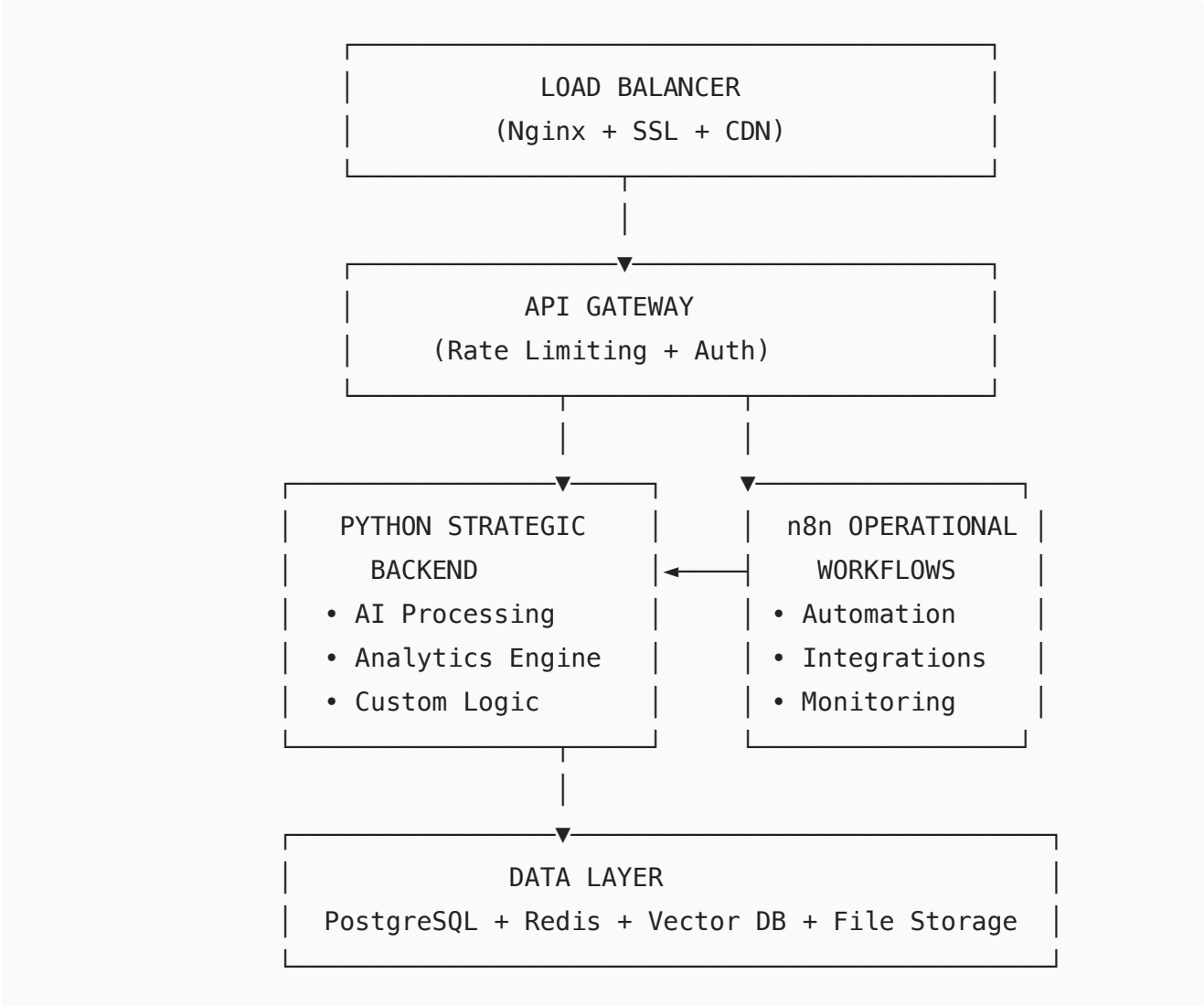
Production-Ready AI Automation Empire with n8n

Complete Infrastructure & Deployment Architecture

🏗️ Production Architecture Overview

Enterprise-Grade Dual Control System:



Production Infrastructure Stack:

```
🌐 FRONTEND LAYER
├── React.js Strategic Dashboard (CEO-level insights)
├── n8n Visual Interface (Operational control)
├── Flutter Mobile App (On-the-go management)
```

```
└── Progressive Web App (Cross-platform access)

🔗 API & INTEGRATION LAYER
├── FastAPI Strategic Backend (Python intelligence)
├── n8n Workflow Engine (Visual automation)
├── WebSocket Real-time Communication
├── GraphQL Advanced Queries
└── REST API External Integrations

🧠 PROCESSING LAYER
├── AI/ML Processing Services (Python)
├── Workflow Orchestration (n8n)
├── Background Task Processing (Celery)
├── Real-time Event Processing (Redis Streams)
└── Voice Processing Pipeline (Custom)

💾 DATA LAYER
├── PostgreSQL (Primary database)
├── Redis (Caching + Sessions + Queues)
├── ChromaDB (Vector embeddings)
├── S3-Compatible Storage (Files + Backups)
└── InfluxDB (Time-series metrics)

🛡️ SECURITY & MONITORING
├── OAuth2 + JWT Authentication
├── End-to-end Encryption
├── Prometheus + Grafana Monitoring
├── ELK Stack Logging
└── Security Scanning & Alerts
```

🐋 Containerized Production Setup

Complete Docker Compose Configuration:

```yaml
# docker-compose.production.yml
version: '3.8'

services:
  # Nginx Load Balancer & SSL Termination
  nginx:
    image: nginx:alpine
    ports:
```

```
      - "80:80"
      - "443:443"
    volumes:
      - ./nginx/nginx.conf:/etc/nginx/nginx.conf
      - ./nginx/ssl:/etc/ssl/certs
      - ./nginx/static:/var/www/static
    depends_on:
      - python-strategic
      - n8n-operational
      - react-dashboard
    restart: unless-stopped
    networks:
      - automation-network

  # Python Strategic Backend
  python-strategic:
    build:
      context: ./python-core
      dockerfile: Dockerfile.production
    environment:
      - ENVIRONMENT=production
      -
DATABASE_URL=postgresql://user:${DB_PASSWORD}@postgres:5432/automation
      - REDIS_URL=redis://redis:6379/0
      - N8N_API_URL=http://n8n-operational:5678
      - VECTOR_DB_URL=http://chromadb:8000
      - OPENAI_API_KEY=${OPENAI_API_KEY}
      - ANTHROPIC_API_KEY=${ANTHROPIC_API_KEY}
      - ELEVENLABS_API_KEY=${ELEVENLABS_API_KEY}
      - JWT_SECRET=${JWT_SECRET}
      - ENCRYPTION_KEY=${ENCRYPTION_KEY}
    volumes:
      - ./data/ai-models:/app/models
      - ./data/analytics:/app/analytics
      - ./data/voice-samples:/app/voice
      - ./logs/python:/app/logs
    depends_on:
      - postgres
      - redis
      - chromadb
    restart: unless-stopped
    deploy:
      resources:
        limits:
          memory: 4G
          cpus: '2.0'
        reservations:
          memory: 2G
          cpus: '1.0'
    networks:
```

```
      - automation-network

  # n8n Operational Workflow Engine
  n8n-operational:
    build:
      context: ./n8n-custom
      dockerfile: Dockerfile.production
    environment:
      - N8N_BASIC_AUTH_ACTIVE=true
      - N8N_BASIC_AUTH_USER=${N8N_USER}
      - N8N_BASIC_AUTH_PASSWORD=${N8N_PASSWORD}
      - DB_TYPE=postgresdb
      - DB_POSTGRESDB_HOST=postgres
      - DB_POSTGRESDB_DATABASE=n8n
      - DB_POSTGRESDB_USER=${DB_USER}
      - DB_POSTGRESDB_PASSWORD=${DB_PASSWORD}
      - N8N_HOST=${DOMAIN_NAME}
      - N8N_PROTOCOL=https
      - N8N_PORT=5678
      - WEBHOOK_URL=https://${DOMAIN_NAME}/n8n-webhook/
      - N8N_METRICS=true
      - N8N_LOG_LEVEL=info
      - PYTHON_API_URL=http://python-strategic:8000
      - EXECUTIONS_TIMEOUT=300
      - EXECUTIONS_TIMEOUT_MAX=600
    volumes:
      - n8n_data:/home/node/.n8n
      - ./n8n-workflows:/home/node/.n8n/workflows
      - ./custom-n8n-nodes:/home/node/.n8n/nodes
      - ./data/n8n-files:/home/node/.n8n/files
      - ./logs/n8n:/home/node/.n8n/logs
    depends_on:
      - postgres
      - redis
    restart: unless-stopped
    deploy:
      resources:
        limits:
          memory: 2G
          cpus: '1.5'
        reservations:
          memory: 1G
          cpus: '0.5'
    networks:
      - automation-network

  # React Strategic Dashboard
  react-dashboard:
    build:
      context: ./react-dashboard
```

```
      dockerfile: Dockerfile.production
    environment:
      - REACT_APP_API_URL=https://${DOMAIN_NAME}/api
      - REACT_APP_N8N_URL=https://${DOMAIN_NAME}/n8n
      - REACT_APP_WS_URL=wss://${DOMAIN_NAME}/ws
      - REACT_APP_SENTRY_DSN=${SENTRY_DSN}
    volumes:
      - react_build:/app/build
    restart: unless-stopped
    networks:
      - automation-network

  # PostgreSQL Primary Database
  postgres:
    image: postgres:15-alpine
    environment:
      - POSTGRES_MULTIPLE_DATABASES=automation,n8n,analytics
      - POSTGRES_USER=${DB_USER}
      - POSTGRES_PASSWORD=${DB_PASSWORD}
      - POSTGRES_DB=automation
    volumes:
      - postgres_data:/var/lib/postgresql/data
      - ./database/init-scripts:/docker-entrypoint-initdb.d
      - ./database/backups:/backups
    ports:
      - "5432:5432"  # For external access if needed
    restart: unless-stopped
    deploy:
      resources:
        limits:
          memory: 2G
          cpus: '1.0'
    networks:
      - automation-network

  # Redis Cache, Sessions & Queues
  redis:
    image: redis:7-alpine
    command: redis-server --appendonly yes --requirepass ${REDIS_PASSWORD}
    volumes:
      - redis_data:/data
      - ./redis/redis.conf:/usr/local/etc/redis/redis.conf
    ports:
      - "6379:6379"  # For external access if needed
    restart: unless-stopped
    deploy:
      resources:
        limits:
          memory: 1G
          cpus: '0.5'
```

```
    networks:
      - automation-network

  # ChromaDB Vector Database
  chromadb:
    image: chromadb/chroma:latest
    environment:
      - CHROMA_SERVER_HOST=0.0.0.0
      - CHROMA_SERVER_PORT=8000
      - PERSIST_DIRECTORY=/chroma/chroma
    volumes:
      - chroma_data:/chroma/chroma
    restart: unless-stopped
    deploy:
      resources:
        limits:
          memory: 1G
          cpus: '0.5'
    networks:
      - automation-network

  # Celery Background Task Worker
  celery-worker:
    build:
      context: ./python-core
      dockerfile: Dockerfile.production
    command: celery -A automation_core worker --loglevel=info --
concurrency=4
    environment:
      - ENVIRONMENT=production
      -
DATABASE_URL=postgresql://user:${DB_PASSWORD}@postgres:5432/automation
      - REDIS_URL=redis://:${REDIS_PASSWORD}@redis:6379/0
      - OPENAI_API_KEY=${OPENAI_API_KEY}
    volumes:
      - ./data/ai-models:/app/models
      - ./data/temp:/app/temp
      - ./logs/celery:/app/logs
    depends_on:
      - postgres
      - redis
    restart: unless-stopped
    deploy:
      replicas: 2
      resources:
        limits:
          memory: 1G
          cpus: '1.0'
    networks:
      - automation-network
```

```yaml
  # Celery Beat Scheduler
  celery-beat:
    build:
      context: ./python-core
      dockerfile: Dockerfile.production
    command: celery -A automation_core beat --loglevel=info
    environment:
      - ENVIRONMENT=production
      -
DATABASE_URL=postgresql://user:${DB_PASSWORD}@postgres:5432/automation
      - REDIS_URL=redis://:${REDIS_PASSWORD}@redis:6379/0
    volumes:
      - ./logs/celery-beat:/app/logs
    depends_on:
      - postgres
      - redis
    restart: unless-stopped
    deploy:
      resources:
        limits:
          memory: 512M
          cpus: '0.5'
    networks:
      - automation-network

  # Prometheus Monitoring
  prometheus:
    image: prom/prometheus:latest
    ports:
      - "9090:9090"
    volumes:
      - ./monitoring/prometheus.yml:/etc/prometheus/prometheus.yml
      - ./monitoring/rules:/etc/prometheus/rules
      - prometheus_data:/prometheus
    command:
      - '--config.file=/etc/prometheus/prometheus.yml'
      - '--storage.tsdb.path=/prometheus'
      - '--web.console.libraries=/etc/prometheus/console_libraries'
      - '--web.console.templates=/etc/prometheus/consoles'
      - '--web.enable-lifecycle'
      - '--web.enable-admin-api'
    restart: unless-stopped
    networks:
      - automation-network

  # Grafana Analytics Dashboard
  grafana:
    image: grafana/grafana:latest
    ports:
```

```
      - "3000:3000"
    environment:
      - GF_SECURITY_ADMIN_USER=${GRAFANA_USER}
      - GF_SECURITY_ADMIN_PASSWORD=${GRAFANA_PASSWORD}
      - GF_INSTALL_PLUGINS=grafana-clock-panel,grafana-simple-json-
datasource
    volumes:
      - grafana_data:/var/lib/grafana
      - ./monitoring/grafana-dashboards:/var/lib/grafana/dashboards
      - ./monitoring/grafana-provisioning:/etc/grafana/provisioning
    restart: unless-stopped
    networks:
      - automation-network

  # Elasticsearch for Advanced Logging
  elasticsearch:
    image: docker.elastic.co/elasticsearch/elasticsearch:8.8.0
    environment:
      - discovery.type=single-node
      - ES_JAVA_OPTS=-Xms1g -Xmx1g
      - xpack.security.enabled=false
    volumes:
      - elasticsearch_data:/usr/share/elasticsearch/data
    ports:
      - "9200:9200"
    restart: unless-stopped
    deploy:
      resources:
        limits:
          memory: 2G
          cpus: '1.0'
    networks:
      - automation-network

  # Kibana for Log Analysis
  kibana:
    image: docker.elastic.co/kibana/kibana:8.8.0
    environment:
      - ELASTICSEARCH_HOSTS=http://elasticsearch:9200
    ports:
      - "5601:5601"
    depends_on:
      - elasticsearch
    restart: unless-stopped
    networks:
      - automation-network

  # MinIO S3-Compatible Storage
  minio:
    image: minio/minio:latest
```

```yaml
    command: server /data --console-address ":9001"
    environment:
      - MINIO_ROOT_USER=${MINIO_ACCESS_KEY}
      - MINIO_ROOT_PASSWORD=${MINIO_SECRET_KEY}
    volumes:
      - minio_data:/data
    ports:
      - "9000:9000"
      - "9001:9001"
    restart: unless-stopped
    networks:
      - automation-network

  # Backup Service
  backup-service:
    build:
      context: ./backup-service
      dockerfile: Dockerfile
    environment:
      -
DATABASE_URL=postgresql://user:${DB_PASSWORD}@postgres:5432/automation
      - BACKUP_SCHEDULE=0 2 * * *  # Daily at 2 AM
      - S3_ENDPOINT=http://minio:9000
      - S3_ACCESS_KEY=${MINIO_ACCESS_KEY}
      - S3_SECRET_KEY=${MINIO_SECRET_KEY}
    volumes:
      - ./data:/app/data
      - ./backups:/app/backups
    depends_on:
      - postgres
      - minio
    restart: unless-stopped
    networks:
      - automation-network

volumes:
  postgres_data:
  redis_data:
  n8n_data:
  chroma_data:
  prometheus_data:
  grafana_data:
  elasticsearch_data:
  minio_data:
  react_build:

networks:
  automation-network:
    driver: bridge
```

⚙️ Advanced Nginx Configuration

**Production