Configuration and Dependency Guide

This document outlines the configuration requirements and dependencies for implementing the remaining components of the Affiliate Matrix system.

System Dependencies

Core Dependencies

Dependency	Version	Purpose
Python	3.10+	Backend development
FastAPI	0.95+	API framework
Pydantic	2.0+	Data validation and settings management
SQLAlchemy	2.0+	ORM for database access
Redis	7.0+	Caching and message broker
PostgreSQL	14.0+	Primary database
Vue.js	3.0+	Frontend framework
Node.js	18.0+	Frontend development

Component-Specific Dependencies

1. Aggregator Connection & API Integration (Steps 1-2)

Dependency	Version	Purpose
Requests	2.28+	HTTP client for API calls
BeautifulSoup4	4.11+	HTML parsing for web scraping
Selenium	4.8+	Browser automation for complex scraping
aiohttp	3.8+	Async HTTP client for parallel requests
tenacity	8.2+	Retry logic for API calls

Dependency	Version	Purpose
httpx	0.24+	HTTP client with timeout handling

2. Key Management (Step 3)

Dependency	Version	Purpose
hvac	1.1+	HashiCorp Vault client
cryptography	40.0+	Cryptographic operations
python-jose	3.3+	JWT handling
passlib	1.7+	Password hashing

3. Master Index & Dynamic Indexing (Steps 4-5)

Dependency	Version	Purpose
Elasticsearch	8.0+	Search and indexing (optional)
elasticsearch-dsl	8.0+	Elasticsearch query DSL
SQLAlchemy	2.0+	ORM for database access
alembic	1.10+	Database migrations
redis	4.5+	Caching
msgpack	1.0+	Efficient serialization

4. Trigger System (Step 7)

Dependency	Version	Purpose
APScheduler	3.10+	Task scheduling
pydantic	2.0+	Rule validation
celery	5.2+	Distributed task queue
redis	4.5+	Message broker

5. Budgeting System (Step 8)

Dependency	Version	Purpose
numpy	1.24+	Numerical operations
pandas	2.0+	Data analysis
scikit-learn	1.2+	Machine learning for optimization
matplotlib	3.7+	Visualization
plotly	5.14+	Interactive visualizations

6. Apex Optimizations (Step 9)

Dependency	Version	Purpose
psutil	5.9+	System monitoring
prometheus-client	0.16+	Metrics collection
kubernetes	26.1+	K3s integration
docker	6.0+	Container management

7. Monitoring System (Step 10)

Dependency	Version	Purpose
prometheus-client	0.16+	Metrics collection
grafana	9.0+	Visualization and dashboards
loki	2.8+	Log aggregation
opentelemetry-api	1.16+	Distributed tracing
opentelemetry-sdk	1.16+	Tracing implementation
sentry-sdk	1.19+	Error tracking

Configuration Requirements

Environment Variables

Create a `.env` file in the project root with the following variables:

```
# Database
DATABASE_URL=postgresql://user:password@localhost:5432/affiliate_matrix
TEST_DATABASE_URL=postgresql://user:password@localhost:5432/
affiliate_matrix_test

# Redis
REDIS_URL=redis://localhost:6379/0
CACHE_TTL=3600

# Security
SECRET_KEY=your-secret-key
API_KEY_ENCRYPTION_KEY=your-encryption-key

# Vault
VAULT_ADDR=http://localhost:8200
VAULT_TOKEN=your-vault-token

# External APIs
OFFERVAULT_API_URL=https://api.offervault.com/v1
AFFILIATEFIX_API_URL=https://api.affiliatefix.com/v1
AFFILIATEPROGRAMS_API_URL=https://api.affiliateprograms.com/v1

# Google Dorking
GOOGLE_API_KEY=your-google-api-key
GOOGLE_CSE_ID=your-custom-search-engine-id
DORKING_RATE_LIMIT=100

# Monitoring
PROMETHEUS_MULTIPROC_DIR=/tmp/prometheus
LOG_LEVEL=INFO
SENTRY_DSN=your-sentry-dsn

# Budgeting
DEFAULT_ALLOCATION_STRATEGY=proportional
DEFAULT_PERFORMANCE_METRIC=roi

# Kubernetes
K3S_KUBECONFIG=/path/to/kubeconfig
```

Configuration Files

1. Database Configuration (`config/database.py`)

```
 from sqlalchemy import create_engine
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker
import os
from dotenv import load_dotenv

load_dotenv()

SQLALCHEMY_DATABASE_URL = os.getenv("DATABASE_URL")

engine = create_engine(SQLALCHEMY_DATABASE_URL)
SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)
Base = declarative_base()

def get_db():
    db = SessionLocal()
    try:
        yield db
    finally:
        db.close()
```

2. Redis Configuration (`config/redis.py`)

```
 import redis
import os
from dotenv import load_dotenv

load_dotenv()

redis_client = redis.Redis.from_url(os.getenv("REDIS_URL"))
cache_ttl = int(os.getenv("CACHE_TTL", 3600))

def get_redis():
    return redis_client
```

3. Vault Configuration (`config/vault.py`)

```
 import hvac
import os
from dotenv import load_dotenv

load_dotenv()
```

```
vault_client = hvac.Client(
    url=os.getenv("VAULT_ADDR"),
    token=os.getenv("VAULT_TOKEN")
)

def get_vault():
    return vault_client
```

4. Logging Configuration (`config/logging.py`)

```
 import logging
import os
from dotenv import load_dotenv

load_dotenv()

log_level = os.getenv("LOG_LEVEL", "INFO")
log_format = "%(asctime)s - %(name)s - %(levelname)s - %(message)s"

logging.basicConfig(
    level=getattr(logging, log_level),
    format=log_format
)

def get_logger(name):
    return logging.getLogger(name)
```

Docker Configuration

Docker Compose (`docker-compose.yml`)

```
 version: '3.8'

services:
  postgres:
    image: postgres:14
    environment:
      POSTGRES_USER: user
      POSTGRES_PASSWORD: password
      POSTGRES_DB: affiliate_matrix
    ports:
      - "5432:5432"
    volumes:
      - postgres_data:/var/lib/postgresql/data

  redis:
    image: redis:7
    ports:
```

```yaml
    - "6379:6379"

  vault:
    image: vault:1.13
    cap_add:
      - IPC_LOCK
    ports:
      - "8200:8200"
    environment:
      VAULT_DEV_ROOT_TOKEN_ID: your-vault-token
      VAULT_DEV_LISTEN_ADDRESS: 0.0.0.0:8200

  elasticsearch:
    image: elasticsearch:8.7.0
    environment:
      - discovery.type=single-node
      - xpack.security.enabled=false
    ports:
      - "9200:9200"

  prometheus:
    image: prom/prometheus:v2.44.0
    ports:
      - "9090:9090"
    volumes:
      - ./config/prometheus.yml:/etc/prometheus/prometheus.yml

  grafana:
    image: grafana/grafana:9.5.2
    ports:
      - "3000:3000"
    environment:
      GF_SECURITY_ADMIN_PASSWORD: admin
    volumes:
      - grafana_data:/var/lib/grafana

  loki:
    image: grafana/loki:2.8.2
    ports:
      - "3100:3100"
    command: -config.file=/etc/loki/local-config.yaml

  api:
    build:
      context: ./backend
    ports:
      - "8000:8000"
    environment:
      - DATABASE_URL=postgresql://user:password@postgres:5432/affiliate_matrix
      - REDIS_URL=redis://redis:6379/0
      - VAULT_ADDR=http://vault:8200
      - ELASTICSEARCH_URL=http://elasticsearch:9200
```

```
    depends_on:
      - postgres
      - redis
      - vault
      - elasticsearch

  frontend:
    build:
      context: ./frontend
    ports:
      - "8080:80"
    depends_on:
      - api

volumes:
  postgres_data:
  grafana_data:
```

Kubernetes Configuration

Deployment Manifest (`k8s/deployment.yaml`)

```
 apiVersion: apps/v1
kind: Deployment
metadata:
  name: affiliate-matrix-api
spec:
  replicas: 3
  selector:
    matchLabels:
      app: affiliate-matrix-api
  template:
    metadata:
      labels:
        app: affiliate-matrix-api
    spec:
      containers:
      - name: api
        image: affiliate-matrix-api:latest
        ports:
        - containerPort: 8000
        env:
        - name: DATABASE_URL
          valueFrom:
            secretKeyRef:
              name: affiliate-matrix-secrets
              key: database-url
        - name: REDIS_URL
          valueFrom:
            secretKeyRef:
```

```
            name: affiliate-matrix-secrets
            key: redis-url
        - name: VAULT_ADDR
          value: http://vault:8200
        resources:
          limits:
            cpu: "1"
            memory: "1Gi"
          requests:
            cpu: "500m"
            memory: "512Mi"
        livenessProbe:
          httpGet:
            path: /health
            port: 8000
          initialDelaySeconds: 30
          periodSeconds: 10
        readinessProbe:
          httpGet:
            path: /health
            port: 8000
          initialDelaySeconds: 5
          periodSeconds: 5
```

Installation and Setup Guide

1. Clone the Repository

```
git clone https://github.com/your-org/affiliate-matrix.git
cd affiliate-matrix
```

2. Set Up Environment Variables

```
cp .env.example .env
# Edit .env with your configuration
```

3. Start Development Environment with Docker

```
docker-compose up -d
```

4. Initialize the Database

```
docker-compose exec api alembic upgrade head
```

5. Initialize Vault

```
docker-compose exec vault vault secrets enable -path=aggregators kv-v2
```

6. Run Backend Tests

```
docker-compose exec api pytest
```

7. Access Services

- API: http://localhost:8000
- API Documentation: http://localhost:8000/docs
- Frontend: http://localhost:8080
- Prometheus: http://localhost:9090
- Grafana: http://localhost:3000
- Vault: http://localhost:8200

Development Workflow

1. Create a feature branch from `main`
2. Implement the feature following the TODO comments
3. Write tests for the new functionality
4. Run the test suite to ensure everything passes
5. Submit a pull request for review

Troubleshooting Common Issues

Database Connection Issues

If you encounter database connection issues:

1. Ensure PostgreSQL is running: `docker-compose ps`
2. Check the database URL in your `.env` file

3. Try connecting manually: `psql postgresql://user:password@localhost:5432/affiliate_matrix`

Redis Connection Issues

If Redis connection fails:

1. Ensure Redis is running: `docker-compose ps`
2. Check the Redis URL in your `.env` file
3. Try connecting manually: `redis-cli -u redis://localhost:6379`

Vault Issues

If Vault integration fails:

1. Ensure Vault is running: `docker-compose ps`
2. Check if Vault is sealed: `docker-compose exec vault vault status`
3. Verify your token: `docker-compose exec vault vault token lookup`

Conclusion

This configuration and dependency guide provides the necessary information to set up the development environment for implementing the remaining components of the Affiliate Matrix system. Follow the installation steps and refer to the component-specific dependencies when implementing each part of the system.

For any questions or issues not covered in this guide, please refer to the project documentation or contact the project maintainers.