Analytics Service Deployment Guide

Overview

This guide covers deploying the Analytics Service to various environments.

Table of Contents

Prerequisites

System Requirements

- **CPU**: 2+ cores (4+ recommended for production)
- **RAM**: 2GB minimum (4GB+ recommended)
- **Storage**: 20GB minimum (depends on data retention)
- **OS**: Linux (Ubuntu 20.04+ recommended)

Software Requirements

- Python 3.11+
- PostgreSQL 15+
- Docker & Docker Compose (for containerized deployment)
- Git

Environment Configuration

Required Environment Variables

```
# Application
PROJECT_NAME="Analytics Service"
PORT=8009
ENVIRONMENT="production"
DEBUG="false"

# Security
SECRET_KEY="<generate-strong-key>"  # openssl rand -hex 32
ACCESS_TOKEN_EXPIRE_MINUTES=30
REFRESH_TOKEN_EXPIRE_DAYS=7

# Database
DATABASE_URL="postgresql+asyncpg://user:password@host:5432/analytics_db"

# Service URLs
PARTNERS_CRM_URL="http://partners-crm:8003"
PROJECTS_URL="http://projects:8004"
SOCIAL_MEDIA_URL="http://social-media:8007"
NOTIFICATION_URL="http://notification:8008"

# Sync Configuration
SYNC_ENABLED="true"
SYNC_INTERVAL_MINUTES=60

# Logging
LOG_LEVEL="INFO"
```

Generate Secrets

```
# Generate SECRET_KEY
openssl rand -hex 32

# Generate strong database password
openssl rand -base64 32
```

Database Setup

PostgreSQL Installation

Ubuntu/Debian:

```
sudo apt update
sudo apt install postgresql-15 postgresql-contrib
```

macOS:

```
brew install postgresql@15
```

Create Database

```
# Connect to PostgreSQL
sudo -u postgres psql

# Create database and user
CREATE DATABASE analytics_db;
CREATE USER analytics_user WITH ENCRYPTED PASSWORD 'your-secure-password';
GRANT ALL PRIVILEGES ON DATABASE analytics_db TO analytics_user;
\q
```

Run Migrations

```
# Activate virtual environment
source venv/bin/activate

# Run migrations
alembic upgrade head
```

Local Deployment

Option 1: Direct Python

1. Clone Repository

```
git clone https://github.com/mission-engadi/analytics-service.git
cd analytics-service
```

2. Create Virtual Environment

```
python -m venv venv
source venv/bin/activate  # Windows: venv\Scripts\activate
```

3. Install Dependencies

```
pip install -r requirements.txt
```

4. Configure Environment

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

5. Run Migrations

```
alembic upgrade head
```

6. Start Service

```
# Using script
./scripts/start.sh

# Or directly
uvicorn app.main:app --host 0.0.0.0 --port 8009
```

Option 2: Using Scripts

```
# Start service
./scripts/start.sh

# Check status
./scripts/status.sh

# Stop service
./scripts/stop.sh

# Restart service
./scripts/restart.sh
```

Docker Deployment

Option 1: Docker Compose (Recommended)

1. Create docker-compose.yml

```yaml
version: '3.8'

services:
  analytics-service:
    image: mission-engadi/analytics-service:latest
    ports:
      - "8009:8009"
    environment:
      - DATABASE_URL=postgresql+asyncpg://postgres:postgres@db:5432/analytics_db
      - SECRET_KEY=${SECRET_KEY}
      - ENVIRONMENT=production
    depends_on:
      - db
    restart: unless-stopped

  db:
    image: postgres:15-alpine
    environment:
      - POSTGRES_DB=analytics_db
      - POSTGRES_PASSWORD=postgres
    volumes:
      - postgres_data:/var/lib/postgresql/data
    restart: unless-stopped

volumes:
  postgres_data:
```

2. Deploy

```
# Start services
docker-compose up -d

# Run migrations
docker-compose exec analytics-service alembic upgrade head

# View logs
docker-compose logs -f

# Stop services
docker-compose down
```

Option 2: Docker Only

1. Build Image

```
docker build -t analytics-service:latest .
```

2. Run Container

```
docker run -d \
  --name analytics-service \
  -p 8009:8009 \
  -e DATABASE_URL="postgresql+asyncpg://..." \
  -e SECRET_KEY="your-secret-key" \
  analytics-service:latest
```

Production Deployment

Systemd Service (Linux)

1. Create Service File

```
sudo nano /etc/systemd/system/analytics-service.service
```

Content:

```
[Unit]
Description=Analytics Service
After=network.target postgresql.service

[Service]
Type=simple
User=ubuntu
WorkingDirectory=/home/ubuntu/analytics-service
Environment="PATH=/home/ubuntu/analytics-service/venv/bin"
ExecStart=/home/ubuntu/analytics-service/venv/bin/uvicorn app.main:app --host 0.0.0.0
--port 8009
Restart=always
RestartSec=10

[Install]
WantedBy=multi-user.target
```

2. Enable and Start

```
# Reload systemd
sudo systemctl daemon-reload

# Enable service
sudo systemctl enable analytics-service

# Start service
sudo systemctl start analytics-service

# Check status
sudo systemctl status analytics-service

# View logs
sudo journalctl -u analytics-service -f
```

Nginx Reverse Proxy

1. Install Nginx

```
sudo apt install nginx
```

2. Configure Nginx

```
sudo nano /etc/nginx/sites-available/analytics-service
```

Content:

```
server {
    listen 80;
    server_name analytics.engadi.org;

    location / {
        proxy_pass http://localhost:8009;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

3. Enable Site

```
sudo ln -s /etc/nginx/sites-available/analytics-service /etc/nginx/sites-enabled/
sudo nginx -t
sudo systemctl reload nginx
```

SSL Certificate (Let's Encrypt)

```
# Install Certbot
sudo apt install certbot python3-certbot-nginx

# Obtain certificate
sudo certbot --nginx -d analytics.engadi.org

# Auto-renewal
sudo certbot renew --dry-run
```

Cloud Deployment

AWS (EC2)

1. Launch EC2 Instance

- AMI: Ubuntu 22.04
- Instance Type: t3.medium (or larger)
- Security Group: Allow ports 22, 80, 443, 8009

2. Install Dependencies

```
sudo apt update
sudo apt install python3.11 python3.11-venv postgresql-client
```

3. Deploy Application (follow Local Deployment steps)

4. Configure RDS (optional)

- Create PostgreSQL RDS instance
- Update DATABASE_URL in .env

DigitalOcean

1. Create Droplet

- Ubuntu 22.04
- 2GB RAM minimum

2. Follow systemd deployment steps

Fly.io

1. Install Fly CLI

```
curl -L https://fly.io/install.sh | sh
```

2. Login

```
fly auth login
```

3. Create App

```
fly launch --name analytics-service
```

4. Create PostgreSQL

```
fly postgres create --name analytics-db
fly postgres attach analytics-db
```

5. Set Secrets

```
fly secrets set SECRET_KEY="your-secret-key"
```

6. Deploy

```
fly deploy
```

Monitoring and Maintenance

Health Checks

Liveness Check:

```
curl http://localhost:8009/api/v1/health
```

Readiness Check:

```
curl http://localhost:8009/api/v1/ready
```

Log Management

View Logs:

```
# Systemd
sudo journalctl -u analytics-service -f

# Docker
docker-compose logs -f analytics-service

# File logs
tail -f /home/ubuntu/analytics-service/logs/analytics_service.log
```

Database Maintenance

Backup:

```
# Create backup
pg_dump -h localhost -U analytics_user analytics_db > backup_$(date +%Y%m%d).sql

# Restore backup
psql -h localhost -U analytics_user analytics_db < backup_20241224.sql
```

Vacuum:

```
psql -h localhost -U analytics_user -d analytics_db -c "VACUUM ANALYZE;"
```

Performance Monitoring

Check Database Connections:

```
SELECT count(*) FROM pg_stat_activity WHERE datname = 'analytics_db';
```

Check Table Sizes:

```
SELECT
    schemaname,
    tablename,
    pg_size_pretty(pg_total_relation_size(schemaname||'.'||tablename)) AS size
FROM pg_tables
WHERE schemaname = 'public'
ORDER BY pg_total_relation_size(schemaname||'.'||tablename) DESC;
```

Troubleshooting

Service Won’t Start

Check logs:

```
sudo journalctl -u analytics-service -n 50
```

Common issues:

1. Port already in use
2. Database connection failed

3. Missing environment variables
4. Permission issues

Database Connection Issues

Test connection:

```
psql -h localhost -U analytics_user -d analytics_db
```

Check PostgreSQL status:

```
sudo systemctl status postgresql
```

High Memory Usage

Check process:

```
ps aux | grep uvicorn
```

Solutions:
1. Reduce worker count
2. Implement pagination
3. Add database connection pooling
4. Scale horizontally

Slow Queries

Enable query logging:

```
ALTER DATABASE analytics_db SET log_min_duration_statement = 1000;
```

Check slow queries:

```
SELECT query, calls, total_time, mean_time
FROM pg_stat_statements
ORDER BY mean_time DESC
LIMIT 10;
```

Security Checklist

- [ ] Use strong SECRET_KEY
- [ ] Enable HTTPS/SSL
- [ ] Configure firewall (allow only necessary ports)
- [ ] Use strong database passwords
- [ ] Enable database SSL
- [ ] Implement rate limiting
- [ ] Regular security updates
- [ ] Monitor logs for suspicious activity

- [ ] Backup database regularly
- [ ] Restrict database access

Support

For deployment support:
- **Documentation**: See README.md and API_DOCUMENTATION.md
- **Issues**: GitHub Issues
- **Email**: support@engadi.org