AI Service - Deployment Guide

Comprehensive guide for deploying the AI Service in various environments.

Table of Contents

Prerequisites

System Requirements

- **OS**: Linux (Ubuntu 20.04+ recommended) or macOS
- **CPU**: 2+ cores
- **RAM**: 4GB minimum, 8GB recommended
- **Storage**: 20GB minimum

Software Requirements

- Python 3.11+
- PostgreSQL 14+
- Docker 20.10+ (for containerized deployment)
- Git

External Services

- Abacus.AI account with API key
- Auth Service running and accessible
- Content Service (optional, for publishing)
- Social Media Service (optional, for social posting)

Local Development

1. Clone Repository

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

2. Create Virtual Environment

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

3. Install Dependencies

```
pip install -r requirements.txt
pip install -r requirements-dev.txt  # For development tools
```

4. Configure Environment

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

Required Variables:

```
DATABASE_URL=postgresql+asyncpg://user:password@localhost:5432/ai_service
SECRET_KEY=your-secret-key-here
JWT_SECRET_KEY=your-jwt-secret-key
ABACUS_API_KEY=your-abacus-api-key
```

5. Setup Database

```
# Create database
createdb ai_service

# Run migrations
alembic upgrade head
```

6. Start Service

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

# Or manually
uvicorn app.main:app --host 0.0.0.0 --port 8010 --reload
```

7. Verify Installation

```
# Check status
./scripts/status.sh

# Test health endpoint
curl http://localhost:8010/api/v1/health

# Access API docs
open http://localhost:8010/docs
```

Docker Deployment

Using Docker Compose (Recommended)

1. Configure environment:

```
cp .env.example .env
# Edit .env
```

2. Start services:

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

3. View logs:

```
docker-compose logs -f ai-service
```

4. Stop services:

```
docker-compose down
```

Using Docker Directly

1. Build image:

```
docker build -t mission-engadi/ai-service:latest .
```

2. Run container:

```
docker run -d \
  --name ai-service \
  -p 8010:8010 \
  -e DATABASE_URL="postgresql+asyncpg://user:pass@host:5432/ai_service" \
  -e ABACUS_API_KEY="your-key" \
  -e SECRET_KEY="your-secret" \
  mission-engadi/ai-service:latest
```

3. Check logs:

```
docker logs -f ai-service
```

Production Deployment

1. Server Setup

Update system:

```
sudo apt update && sudo apt upgrade -y
```

Install dependencies:

```
sudo apt install -y python3.11 python3.11-venv postgresql nginx certbot
```

2. Application Setup

Create application user:

```
sudo useradd -m -s /bin/bash aiservice
sudo su - aiservice
```

Clone and setup:

```
git clone https://github.com/mission-engadi/ai-service.git
cd ai-service
python3.11 -m venv venv
source venv/bin/activate
pip install -r requirements.txt
```

Configure environment:

```
cp .env.example .env
vim .env
```

Production .env:

```
ENVIRONMENT=production
DEBUG=false
DATABASE_URL=postgresql+asyncpg://aiservice:password@localhost:5432/ai_service_prod
SECRET_KEY=<generate-strong-key>
JWT_SECRET_KEY=<generate-strong-key>
ABACUS_API_KEY=<your-api-key>
LOG_LEVEL=INFO
```

3. Database Setup

```
sudo -u postgres psql
```

```
CREATE DATABASE ai_service_prod;
CREATE USER aiservice WITH ENCRYPTED PASSWORD 'strong_password';
GRANT ALL PRIVILEGES ON DATABASE ai_service_prod TO aiservice;
\q
```

Run migrations:

```
alembic upgrade head
```

4. Systemd Service

Create service file:

```
sudo vim /etc/systemd/system/ai-service.service
```

```
[Unit]
Description=AI Service - Mission Engadi
After=network.target postgresql.service

[Service]
Type=simple
User=aiservice
Group=aiservice
WorkingDirectory=/home/aiservice/ai-service
Environment="PATH=/home/aiservice/ai-service/venv/bin"
EnvironmentFile=/home/aiservice/ai-service/.env
ExecStart=/home/aiservice/ai-service/venv/bin/uvicorn app.main:app --host 0.0.0.0 --
port 8010 --workers 4
Restart=always
RestartSec=10

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

Enable and start:

```
sudo systemctl daemon-reload
sudo systemctl enable ai-service
sudo systemctl start ai-service
sudo systemctl status ai-service
```

5. Nginx Reverse Proxy

Create Nginx config:

```
sudo vim /etc/nginx/sites-available/ai-service
```

```
upstream ai_service {
    server 127.0.0.1:8010;
}

server {
    listen 80;
    server_name ai.mission-engadi.org;

    location / {
        proxy_pass http://ai_service;
        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;
        proxy_read_timeout 300;
        proxy_connect_timeout 300;
    }
}
```

Enable site:

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

6. SSL Certificate

```
sudo certbot --nginx -d ai.mission-engadi.org
```

Kubernetes Deployment

1. ConfigMap

configmap.yaml:

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: ai-service-config
  namespace: mission-engadi
data:
  ENVIRONMENT: "production"
  LOG_LEVEL: "INFO"
  SERVICE_PORT: "8010"
```

2. Secrets

```
kubectl create secret generic ai-service-secrets \
  --from-literal=DATABASE_URL=postgresql+asyncpg://... \
  --from-literal=SECRET_KEY=... \
  --from-literal=JWT_SECRET_KEY=... \
  --from-literal=ABACUS_API_KEY=... \
  -n mission-engadi
```

3. Deployment

deployment.yaml:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ai-service
  namespace: mission-engadi
spec:
  replicas: 3
  selector:
    matchLabels:
      app: ai-service
  template:
    metadata:
      labels:
        app: ai-service
    spec:
      containers:
      - name: ai-service
        image: mission-engadi/ai-service:latest
        ports:
        - containerPort: 8010
        envFrom:
        - configMapRef:
            name: ai-service-config
        - secretRef:
            name: ai-service-secrets
        resources:
          requests:
            memory: "512Mi"
            cpu: "250m"
          limits:
            memory: "2Gi"
            cpu: "1000m"
        livenessProbe:
          httpGet:
            path: /api/v1/health
            port: 8010
          initialDelaySeconds: 30
          periodSeconds: 10
        readinessProbe:
          httpGet:
            path: /api/v1/ready
            port: 8010
          initialDelaySeconds: 10
          periodSeconds: 5
```

4. Service

service.yaml:

```yaml
apiVersion: v1
kind: Service
metadata:
  name: ai-service
  namespace: mission-engadi
spec:
  selector:
    app: ai-service
  ports:
  - port: 80
    targetPort: 8010
  type: ClusterIP
```

5. Ingress

ingress.yaml:

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ai-service-ingress
  namespace: mission-engadi
  annotations:
    cert-manager.io/cluster-issuer: "letsencrypt-prod"
spec:
  tls:
  - hosts:
    - ai.mission-engadi.org
    secretName: ai-service-tls
  rules:
  - host: ai.mission-engadi.org
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: ai-service
            port:
              number: 80
```

6. Deploy

```
kubectl apply -f k8s/
kubectl get pods -n mission-engadi
kubectl logs -f deployment/ai-service -n mission-engadi
```

Environment Configuration

Development

```
ENVIRONMENT=development
DEBUG=true
LOG_LEVEL=DEBUG
DATABASE_URL=postgresql+asyncpg://user:pass@localhost:5432/ai_service_dev
```

Staging

```
ENVIRONMENT=staging
DEBUG=false
LOG_LEVEL=INFO
DATABASE_URL=postgresql+asyncpg://user:pass@staging-db:5432/ai_service_staging
```

Production

```
ENVIRONMENT=production
DEBUG=false
LOG_LEVEL=WARNING
DATABASE_URL=postgresql+asyncpg://user:pass@prod-db:5432/ai_service_prod
```

Monitoring & Logging

Application Logs

Using systemd:

```
sudo journalctl -u ai-service -f
```

Using Docker:

```
docker logs -f ai-service
```

Health Monitoring

```
# Health check
curl https://ai.mission-engadi.org/api/v1/health

# Readiness check
curl https://ai.mission-engadi.org/api/v1/ready
```

Scaling

Horizontal Scaling

Docker Compose:

```
docker-compose up -d --scale ai-service=3
```

Kubernetes:

```
kubectl scale deployment ai-service --replicas=5 -n mission-engadi
```

Vertical Scaling

Update resources in deployment.yaml and apply.

Troubleshooting

Service Won't Start

Check logs:

```
sudo journalctl -u ai-service -n 100
```

Common issues:
- Database connection: Verify DATABASE_URL
- Port in use: Check if port 8010 is available
- Missing dependencies: Reinstall requirements

Database Connection Errors

```
# Test connection
psql $DATABASE_URL

# Check PostgreSQL status
sudo systemctl status postgresql
```

High Memory Usage

- Reduce worker count
- Enable database connection pooling
- Implement request rate limiting

Last Updated: December 2024
Version: 1.0.0