# Gateway Service Deployment Guide

> Production deployment guide for the Mission Engadi Gateway Service

## Table of Contents



---

## Overview

This guide covers deploying the Gateway Service in various environments, from local development to production cloud deployments.

### Deployment Options

- **Local Development**: Docker Compose
- **Container Orchestration**: Kubernetes, Docker Swarm
- **Cloud Platforms**: AWS, GCP, Azure, Fly.io
- **Serverless**: AWS Lambda, Google Cloud Functions (with adapters)

---

## Prerequisites

### Required

- **Python**: 3.11 or higher
- **PostgreSQL**: 15 or higher
- **Docker**: 20.10 or higher (for containerized deployment)
- **Git**: For version control

### Recommended

- **Redis**: 7 or higher (for caching and rate limiting)
- **Load Balancer**: For production deployments
- **Monitoring**: Prometheus, Grafana, or cloud-native solutions
- **Log Aggregation**: ELK stack, CloudWatch, or similar

# Local Development

## Quick Start with Scripts

```
# Clone repository
git clone https://github.com/mission-engadi/gateway_service.git
cd gateway_service

# Start service
./start.sh

# Check status
./status.sh

# View logs
tail -f uvicorn.log

# Stop service
./stop.sh

# Restart service
./restart.sh
```

## Manual Setup

1. **Create virtual environment**
   ```
   bash
     python3 -m venv venv
     source venv/bin/activate
   ```

2. **Install dependencies**
   ```
   bash
     pip install -r requirements.txt
   ```

3. **Configure environment**
   ```
   bash
     cp .env.example .env
     # Edit .env with your configuration
   ```

4. **Start PostgreSQL**
   ```
   bash
     docker run -d \
       --name gateway-postgres \
       -p 5432:5432 \
       -e POSTGRES_PASSWORD=postgres \
       -e POSTGRES_DB=gateway_db \
       postgres:15-alpine
   ```

5. **Run migrations**
   ```
   bash
     alembic upgrade head
   ```

6. **Start service**
   `bash`
   `uvicorn app.main:app --reload --port 8000`

---

# Docker Deployment

## Using Docker Compose

### docker-compose.yml

```yaml
version: '3.8'

services:
  gateway:
    build: .
    ports:
      - "8000:8000"
    environment:
      - DATABASE_URL=postgresql+asyncpg://postgres:postgres@postgres:5432/gateway_db
      - REDIS_URL=redis://redis:6379/0
      - SECRET_KEY=${SECRET_KEY}
      - ENVIRONMENT=production
    depends_on:
      - postgres
      - redis
    restart: unless-stopped
    healthcheck:
      test: ["CMD", "curl", "-f", "http://localhost:8000/api/v1/health"]
      interval: 30s
      timeout: 10s
      retries: 3

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

  redis:
    image: redis:7-alpine
    restart: unless-stopped

volumes:
  postgres_data:
```

### Deployment Commands

```
# Build and start
docker-compose up -d --build

# View logs
docker-compose logs -f gateway

# Scale gateway instances
docker-compose up -d --scale gateway=3

# Stop services
docker-compose down

# Remove volumes (CAREFUL!)
docker-compose down -v
```

## Using Docker Directly

```
# Build image
docker build -t gateway-service:latest .

# Run container
docker run -d \
  --name gateway \
  -p 8000:8000 \
  -e DATABASE_URL=postgresql://... \
  -e SECRET_KEY=your-secret-key \
  gateway-service:latest

# View logs
docker logs -f gateway

# Stop container
docker stop gateway

# Remove container
docker rm gateway
```

---

# Kubernetes Deployment

## Deployment Manifests

### deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: gateway-service
  labels:
    app: gateway-service
spec:
  replicas: 3
  selector:
    matchLabels:
      app: gateway-service
  template:
    metadata:
      labels:
        app: gateway-service
    spec:
      containers:
      - name: gateway
        image: mission-engadi/gateway-service:latest
        ports:
        - containerPort: 8000
        env:
        - name: DATABASE_URL
          valueFrom:
            secretKeyRef:
              name: gateway-secrets
              key: database-url
        - name: SECRET_KEY
          valueFrom:
            secretKeyRef:
              name: gateway-secrets
              key: secret-key
        - name: ENVIRONMENT
          value: "production"
        resources:
          requests:
            memory: "256Mi"
            cpu: "250m"
          limits:
            memory: "512Mi"
            cpu: "500m"
        livenessProbe:
          httpGet:
            path: /api/v1/live
            port: 8000
          initialDelaySeconds: 30
          periodSeconds: 10
        readinessProbe:
          httpGet:
            path: /api/v1/ready
            port: 8000
          initialDelaySeconds: 5
          periodSeconds: 5
```

### service.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
  name: gateway-service
spec:
  selector:
    app: gateway-service
  ports:
  - protocol: TCP
    port: 80
    targetPort: 8000
  type: LoadBalancer
```

### secrets.yaml

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: gateway-secrets
type: Opaque
stringData:
  database-url: "postgresql+asyncpg://user:password@postgres:5432/gateway_db"
  secret-key: "your-secret-key-here"
```

### ingress.yaml

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

### Deployment Commands

```
# Create namespace
kubectl create namespace mission-engadi

# Apply secrets
kubectl apply -f secrets.yaml -n mission-engadi

# Deploy application
kubectl apply -f deployment.yaml -n mission-engadi
kubectl apply -f service.yaml -n mission-engadi
kubectl apply -f ingress.yaml -n mission-engadi

# Check deployment
kubectl get pods -n mission-engadi
kubectl get svc -n mission-engadi

# View logs
kubectl logs -f deployment/gateway-service -n mission-engadi

# Scale deployment
kubectl scale deployment gateway-service --replicas=5 -n mission-engadi

# Update deployment
kubectl set image deployment/gateway-service \
  gateway=mission-engadi/gateway-service:v1.1.0 \
  -n mission-engadi

# Rollback deployment
kubectl rollout undo deployment/gateway-service -n mission-engadi
```

---

# Cloud Platform Deployment

## AWS (Elastic Beanstalk)

```
# Install EB CLI
pip install awsebcli

# Initialize EB application
eb init -p python-3.11 gateway-service

# Create environment
eb create gateway-prod

# Set environment variables
eb setenv \
  SECRET_KEY=your-secret-key \
  DATABASE_URL=postgresql://... \
  ENVIRONMENT=production

# Deploy
eb deploy

# Open in browser
eb open

# View logs
eb logs
```

## Google Cloud Platform (Cloud Run)

```
# Build and push to Container Registry
gcloud builds submit --tag gcr.io/PROJECT_ID/gateway-service

# Deploy to Cloud Run
gcloud run deploy gateway-service \
  --image gcr.io/PROJECT_ID/gateway-service \
  --platform managed \
  --region us-central1 \
  --allow-unauthenticated \
  --set-env-vars="SECRET_KEY=your-key,DATABASE_URL=postgresql://..."

# View logs
gcloud run logs read gateway-service
```

### Azure (Container Instances)

```
# Login to Azure
az login

# Create resource group
az group create --name gateway-rg --location eastus

# Create container instance
az container create \
  --resource-group gateway-rg \
  --name gateway-service \
  --image mission-engadi/gateway-service:latest \
  --dns-name-label gateway-engadi \
  --ports 8000 \
  --environment-variables \
    SECRET_KEY=your-key \
    DATABASE_URL=postgresql://...

# View logs
az container logs --resource-group gateway-rg --name gateway-service
```

### Fly.io

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

# Login
fly auth login

# Launch app
fly launch --name gateway-service

# Set secrets
fly secrets set \
  SECRET_KEY=your-secret-key \
  DATABASE_URL=postgresql://...

# Deploy
fly deploy

# Scale
fly scale count 3

# View logs
fly logs
```

---

# Configuration

## Environment Variables

### Required

```
# Application
SECRET_KEY="your-secret-key-here"  # Generate with: openssl rand -hex 32
DATABASE_URL="postgresql+asyncpg://user:pass@host:5432/db"

# Environment
ENVIRONMENT="production"  # production, staging, development
DEBUG="false"
```

### Optional

```
# Redis
REDIS_URL="redis://localhost:6379/0"

# CORS
CORS_ORIGINS="https://app.engadi.org,https://engadi.org"

# Rate Limiting
DEFAULT_RATE_LIMIT="100"
DEFAULT_RATE_WINDOW="60"

# Monitoring
DATADOG_API_KEY="your-datadog-key"
SENTRY_DSN="your-sentry-dsn"

# Logging
LOG_LEVEL="INFO"  # DEBUG, INFO, WARNING, ERROR, CRITICAL
```

## Database Setup

### PostgreSQL

```
-- Create database
CREATE DATABASE gateway_db;

-- Create user
CREATE USER gateway_user WITH PASSWORD 'secure_password';

-- Grant permissions
GRANT ALL PRIVILEGES ON DATABASE gateway_db TO gateway_user;
```

### Run Migrations

```
# Upgrade to latest version
alembic upgrade head

# Check current version
alembic current

# Downgrade if needed
alembic downgrade -1
```

# Monitoring & Observability

## Health Checks

```
# Liveness probe
curl http://localhost:8000/api/v1/live

# Readiness probe
curl http://localhost:8000/api/v1/ready

# Health check
curl http://localhost:8000/api/v1/health
```

## Logging

### Structured Logging

Production logs are in JSON format:

```
{
  "timestamp": "2024-12-25T00:00:00Z",
  "level": "INFO",
  "logger": "app.services.proxy",
  "message": "Request proxied successfully",
  "trace_id": "abc123",
  "user_id": "user456",
  "status_code": 200,
  "response_time": 0.15
}
```

### Log Aggregation

**Docker Compose**:

```
docker-compose logs -f gateway
```

**Kubernetes**:

```
kubectl logs -f deployment/gateway-service -n mission-engadi
```

**Cloud Platforms**:
- AWS: CloudWatch Logs
- GCP: Cloud Logging
- Azure: Azure Monitor

### Metrics

#### Prometheus Integration

```yaml
# prometheus.yml
scrape_configs:
  - job_name: 'gateway'
    static_configs:
      - targets: ['gateway:8000']
```

#### Key Metrics

- **Request rate**: requests per second
- **Response time**: p50, p95, p99 latencies
- **Error rate**: 4xx and 5xx responses
- **Active connections**: current connections
- **Service health**: backend service status

---

## Scaling

### Horizontal Scaling

#### Docker Compose

```bash
docker-compose up -d --scale gateway=5
```

#### Kubernetes

```bash
# Manual scaling
kubectl scale deployment gateway-service --replicas=5

# Auto-scaling
kubectl autoscale deployment gateway-service \
  --cpu-percent=70 \
  --min=3 \
  --max=10
```

### Vertical Scaling

Update resource limits:

```yaml
resources:
  requests:
    memory: "512Mi"
    cpu: "500m"
  limits:
    memory: "1Gi"
    cpu: "1000m"
```

### Load Balancing

- **Docker Compose**: Use Nginx or HAProxy
- **Kubernetes**: Built-in service load balancing

- **Cloud**: Use cloud load balancers (ALB, Cloud Load Balancing, etc.)

---

# Troubleshooting

## Service Won't Start

1. **Check logs**:
   ```bash
   docker logs gateway
   kubectl logs deployment/gateway-service
   ```

2. **Verify environment variables**:
   ```bash
   docker exec gateway env
   ```

3. **Test database connection**:
   ```bash
   docker exec gateway python -c "from app.db.session import engine; print('Connected')"
   ```

## High Response Times

1. **Check resource usage**:
   ```bash
   docker stats gateway
   kubectl top pods
   ```

2. **Review slow queries**:
   ```bash
   # Check PostgreSQL slow query log
   docker exec postgres psql -U postgres -c "SELECT * FROM pg_stat_activity WHERE state = 'active';"
   ```

3. **Scale horizontally**:
   ```bash
   docker-compose up -d --scale gateway=3
   ```

## Database Connection Issues

1. **Verify DATABASE_URL**:
   ```bash
   echo $DATABASE_URL
   ```

2. **Test connection**:
   ```bash
   psql $DATABASE_URL
   ```

3. **Check connection pool**:
   - Increase pool size in config
   - Check for connection leaks

### Memory Leaks

1. **Monitor memory usage**:
   ```
   bash
   docker stats gateway
   ```
2. **Enable memory profiling**:
   ```
   python
   # In app/main.py
   import tracemalloc
   tracemalloc.start()
   ```
3. **Restart service periodically** (temporary fix)

---

## Production Checklist

### Security

- [ ] Strong SECRET_KEY generated
- [ ] HTTPS enabled (TLS certificates)
- [ ] Database credentials secured
- [ ] Environment variables not in code
- [ ] CORS origins configured
- [ ] Rate limiting enabled
- [ ] Security headers configured

### Performance

- [ ] Database connection pooling configured
- [ ] Redis caching enabled
- [ ] CDN configured for static assets
- [ ] Compression enabled
- [ ] Health checks configured

### Monitoring

- [ ] Logging configured
- [ ] Metrics collection enabled
- [ ] Alerts configured
- [ ] Health check endpoints tested
- [ ] Error tracking enabled (Sentry)

### Backup & Recovery

- [ ] Database backups automated
- [ ] Disaster recovery plan documented
- [ ] Rollback procedure tested
- [ ] Data retention policy defined

---

## Support

- **Documentation**: docs.engadi.org (https://docs.engadi.org)
- **Issues**: GitHub Issues (https://github.com/mission-engadi/gateway_service/issues)
- **Email**: support@engadi.org