🚀 Deployment Guide

This comprehensive guide covers all aspects of deploying and managing the Agent Orchestration Operations system.

📋 Table of Contents

🔧 Prerequisites

System Requirements

- **Operating System:** Linux (Ubuntu 20.04+ recommended)
- **Memory:** Minimum 4GB RAM, 8GB+ recommended
- **Storage:** Minimum 20GB free space
- **Network:** Stable internet connection
- **Permissions:** sudo access for system-level operations

Required Tools

```
# Install required tools
sudo apt-get update
sudo apt-get install -y git curl wget unzip

# Install Docker (if using containerized deployment)
curl -fsSL https://get.docker.com -o get-docker.sh
sudo sh get-docker.sh

# Install Docker Compose
sudo curl -L "https://github.com/docker/compose/releases/latest/download/docker-com-
pose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
```

Access Requirements

- GitHub repository access
- Required API keys and tokens
- Environment-specific credentials
- Network access to deployment targets

🌍 Environment Setup

Development Environment

```
# Clone the repository
git clone https://github.com/Empire325Marketing/agent-orchestration-ops.git
cd agent-orchestration-ops

# Set up development environment
cp .env.example .env.development
# Edit .env.development with your configuration

# Install dependencies
./scripts/setup-dev.sh
```

Staging Environment

```
# Set up staging environment
cp .env.example .env.staging
# Configure staging-specific settings

# Deploy to staging
./scripts/deploy-staging.sh
```

Production Environment

```
# Set up production environment
cp .env.example .env.production
# Configure production settings with security considerations

# Deploy to production (requires approval)
./scripts/deploy-production.sh
```

🚀 Deployment Process

Automated Deployment (Recommended)

The system uses GitHub Actions for automated deployment:

1. **Trigger Deployment:**
   ```bash
   # Push to main branch triggers automatic deployment
   git push origin main

# Or trigger manual deployment
gh workflow run "🚀 Continuous Deployment" –ref main
```

1. **Monitor Deployment:**
   - Check GitHub Actions tab for deployment status
   - Monitor logs for any issues
   - Verify health checks pass

2. **Verify Deployment:**
   ```bash
   # Check application status
   curl -f https://your-domain.com/health

# Verify all services are running
./scripts/verify-deployment.sh
```

Manual Deployment

For manual deployment or troubleshooting:

```
# 1. Prepare deployment
./scripts/pre-deployment-checks.sh

# 2. Build application
./scripts/build.sh

# 3. Deploy to target environment
./scripts/deploy.sh --environment production

# 4. Run post-deployment checks
./scripts/post-deployment-checks.sh
```

⚙️ Configuration

Environment Variables

Create environment-specific configuration files:

```
# .env.production
NODE_ENV=production
DATABASE_URL=postgresql://user:pass@host:5432/db
REDIS_URL=redis://host:6379
API_KEY=your-api-key
SECRET_KEY=your-secret-key

# Security settings
CORS_ORIGIN=https://your-domain.com
RATE_LIMIT_MAX=1000
SESSION_TIMEOUT=3600

# Monitoring settings
LOG_LEVEL=info
METRICS_ENABLED=true
HEALTH_CHECK_INTERVAL=30
```

Database Configuration

```
-- Create database and user
CREATE DATABASE agent_orchestration;
CREATE USER app_user WITH PASSWORD 'secure_password';
GRANT ALL PRIVILEGES ON DATABASE agent_orchestration TO app_user;

-- Run migrations
./scripts/migrate.sh
```

Load Balancer Configuration

```
# nginx.conf
upstream app_servers {
    server app1.internal:3000;
    server app2.internal:3000;
    server app3.internal:3000;
}

server {
    listen 80;
    server_name your-domain.com;
    return 301 https://$server_name$request_uri;
}

server {
    listen 443 ssl http2;
    server_name your-domain.com;

    ssl_certificate /path/to/cert.pem;
    ssl_certificate_key /path/to/key.pem;

    location / {
        proxy_pass http://app_servers;
        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;
    }

    location /health {
        access_log off;
        proxy_pass http://app_servers;
    }
}
```

📊 Monitoring & Maintenance

Health Monitoring

The system includes comprehensive health monitoring:

```
# Check system health
curl https://your-domain.com/health

# Expected response:
{
  "status": "healthy",
  "timestamp": "2024-01-01T00:00:00Z",
  "services": {
    "database": "healthy",
    "redis": "healthy",
    "external_apis": "healthy"
  },
  "metrics": {
    "response_time": 150,
    "memory_usage": 65,
    "cpu_usage": 45
  }
}
```

Performance Monitoring

```
# View performance metrics
./scripts/performance-report.sh

# Monitor resource usage
./scripts/resource-monitor.sh

# Generate performance dashboard
./scripts/generate-dashboard.sh
```

Log Management

```
# View application logs
./scripts/view-logs.sh --environment production

# Search logs
./scripts/search-logs.sh --query "error" --since "1h"

# Archive old logs
./scripts/archive-logs.sh --older-than "30d"
```

🔧 Troubleshooting

Common Issues

Deployment Failures

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

# View deployment logs
./scripts/deployment-logs.sh

# Retry failed deployment
./scripts/retry-deployment.sh
```

Performance Issues

```
# Analyze performance bottlenecks
./scripts/performance-analysis.sh

# Check resource usage
./scripts/resource-check.sh

# Optimize database queries
./scripts/query-optimization.sh
```

Security Issues

```
# Run security scan
./scripts/security-scan.sh

# Check for vulnerabilities
./scripts/vulnerability-check.sh

# Update security patches
./scripts/security-update.sh
```

Diagnostic Commands

```
# System diagnostics
./scripts/system-diagnostics.sh

# Network connectivity test
./scripts/network-test.sh

# Database connectivity test
./scripts/db-test.sh

# API endpoint test
./scripts/api-test.sh
```

🔄 Rollback Procedures

Automatic Rollback

The system includes automatic rollback on deployment failure:

```
# Rollback is triggered automatically if:
# - Health checks fail after deployment
# - Critical errors are detected
# - Performance degrades significantly
```

Manual Rollback

```
# Rollback to previous version
./scripts/rollback.sh --version previous

# Rollback to specific version
./scripts/rollback.sh --version v1.2.3

# Emergency rollback (fastest)
./scripts/emergency-rollback.sh
```

Rollback Verification

```
# Verify rollback success
./scripts/verify-rollback.sh

# Check system health after rollback
./scripts/post-rollback-health-check.sh

# Generate rollback report
./scripts/rollback-report.sh
```

🛡️ Security Considerations

Deployment Security

- Use encrypted connections (HTTPS/TLS)
- Implement proper authentication and authorization
- Regular security updates and patches
- Network segmentation and firewalls
- Secure credential management

Access Control

```
# Set up proper file permissions
chmod 600 .env.*
chmod 700 scripts/
chmod 755 public/

# Configure user access
sudo usermod -aG docker deploy-user
sudo usermod -aG sudo deploy-user
```

Secrets Management

```
# Use environment variables for secrets
export DATABASE_PASSWORD=$(vault kv get -field=password secret/db)
export API_KEY=$(vault kv get -field=key secret/api)

# Or use encrypted files
gpg --decrypt secrets.gpg > .env.production
```

📈 Performance Optimization

Database Optimization

```sql
-- Create indexes for frequently queried columns
CREATE INDEX idx_user_email ON users(email);
CREATE INDEX idx_created_at ON logs(created_at);

-- Optimize queries
EXPLAIN ANALYZE SELECT * FROM users WHERE email = 'user@example.com';
```

Caching Strategy

```
# Configure Redis caching
redis-cli CONFIG SET maxmemory 2gb
redis-cli CONFIG SET maxmemory-policy allkeys-lru

# Monitor cache performance
redis-cli INFO memory
redis-cli INFO stats
```

Load Testing

```
# Run load tests before deployment
./scripts/load-test.sh --concurrent 100 --duration 300

# Monitor performance during load test
./scripts/monitor-load-test.sh
```

📚 Additional Resources

- API Documentation (./API_DOCUMENTATION.md)
- Security Guide (./SECURITY_GUIDE.md)
- Monitoring Guide (./MONITORING_GUIDE.md)
- Troubleshooting Guide (./TROUBLESHOOTING_GUIDE.md)

🆘 Support

For deployment issues or questions:

1. Check the troubleshooting section
2. Review deployment logs
3. Contact the development team
4. Create an issue in the repository

This deployment guide is maintained by the Agent Orchestration Operations team. Last updated: $(date)