🚀 Production Deployment Checklist

Pre-Deployment Checklist

✅ Security

- [ ] **Secret Key**: Generate secure 32+ character `SECRET_KEY`

```bash
python -c "import secrets; print(secrets.token_urlsafe(32))"
```

- [ ] **Environment Variables**: All sensitive data in environment variables
- [ ] **Database Password**: Strong password (16+ characters, mixed case, numbers, symbols)
- [ ] **API Keys**: All API keys configured (Anthropic, SerpAPI, YouTube)
- [ ] **HTTPS**: SSL/TLS certificates configured
- [ ] **CORS**: Whitelist only trusted domains
- [ ] **Allowed Hosts**: Remove wildcard, specify exact domains
- [ ] **Rate Limiting**: Enabled and configured appropriately
- [ ] **SQL Injection**: Parameterized queries used everywhere
- [ ] **Input Validation**: All endpoints validate input
- [ ] **File Permissions**: Logs directory writable, code read-only

✅ Database

- [ ] **Connection Pool**: Configured (min: 2, max: 10-20)
- [ ] **Migrations**: All migrations applied
- [ ] **Indexes**: Performance-critical indexes created
- [ ] **Backup**: Automated backup system in place
- [ ] **Monitoring**: Query performance monitoring enabled
- [ ] **Timeouts**: Statement timeout set (30s default)
- [ ] **Connection Limits**: Max connections configured properly
- [ ] **SSL Mode**: Database SSL enabled if supported

✅ Redis

- [ ] **Password**: Redis password set if exposed
- [ ] **Max Memory**: Memory limit configured
- [ ] **Persistence**: AOF or RDB enabled
- [ ] **Eviction Policy**: Set to `allkeys-lru` or similar
- [ ] **Monitoring**: Redis monitoring enabled

✅ AI/ML Services

- [ ] **Anthropic API Key**: Valid and tested
- [ ] **SerpAPI Key**: Valid with sufficient quota
- [ ] **YouTube API**: OAuth credentials configured
- [ ] **Circuit Breakers**: Configured for all external APIs
- [ ] **Fallback Models**: Multiple models configured
- [ ] **Retry Logic**: Exponential backoff implemented
- [ ] **Timeout Settings**: Reasonable timeouts (30-60s)
- [ ] **Cost Monitoring**: API usage tracking enabled

✅ Application Configuration

- [ ] **Environment**: Set to `production`
- [ ] **Debug**: Set to `false`
- [ ] **Workers**: Configured based on CPU cores (2-4 per core)
- [ ] **Logging**: Log level set to `INFO` or `WARNING`
- [ ] **Log Format**: JSON for production
- [ ] **Sentry**: Error tracking configured
- [ ] **Metrics**: Prometheus/monitoring enabled
- [ ] **Health Checks**: All health endpoints working

✅ Performance

- [ ] **Connection Pooling**: Database and Redis pools configured
- [ ] **Caching**: Redis caching enabled
- [ ] **Compression**: GZip compression enabled
- [ ] **Static Files**: Served via CDN or nginx
- [ ] **Query Optimization**: N+1 queries eliminated
- [ ] **Batch Operations**: Batch processing for bulk operations
- [ ] **Async Operations**: Background tasks for heavy operations

✅ Monitoring & Observability

- [ ] **Logging**: Structured logging implemented
- [ ] **Metrics**: Key metrics being collected
- [ ] **Alerts**: Critical alerts configured
- [ ] **Dashboards**: Monitoring dashboards created
- [ ] **Error Tracking**: Sentry or similar configured
- [ ] **APM**: Application performance monitoring (optional)

- [ ] **Uptime Monitoring**: External uptime checks

✅ Reliability

- [ ] **Circuit Breakers**: Enabled for external services
- [ ] **Retry Logic**: Exponential backoff implemented
- [ ] **Timeouts**: All external calls have timeouts
- [ ] **Graceful Degradation**: Fallbacks for AI services
- [ ] **Health Checks**: Liveness and readiness probes
- [ ] **Auto-Restart**: Process manager configured (systemd, supervisor)
- [ ] **Rate Limiting**: Per-user and global limits

✅ Testing

- [ ] **Unit Tests**: Core functionality tested
- [ ] **Integration Tests**: API endpoints tested
- [ ] **Load Testing**: Performance under load verified
- [ ] **Security Scan**: Vulnerability scan completed
- [ ] **Dependency Audit**: `pip-audit` or `safety` check passed
- [ ] **Manual Testing**: Critical user flows verified

✅ Documentation

- [ ] **README**: Updated with deployment instructions
- [ ] **API Docs**: Swagger/ReDoc enabled
- [ ] **Configuration Guide**: All settings documented
- [ ] **Troubleshooting Guide**: Common issues documented
- [ ] **Runbook**: Operations procedures documented

✅ Infrastructure

- [ ] **Domain**: DNS configured and propagated
- [ ] **SSL Certificate**: Valid and auto-renewing
- [ ] **Firewall**: Only necessary ports open
- [ ] **Reverse Proxy**: Nginx or similar configured
- [ ] **Load Balancer**: If using multiple instances
- [ ] **CDN**: For static assets (optional)
- [ ] **Backup**: Automated backups configured

✅ Compliance

- [ ] **GDPR**: Data privacy measures implemented

- [ ] **Data Retention**: Policies configured
- [ ] **Audit Logging**: User actions logged
- [ ] **Terms of Service**: Legal documentation ready
- [ ] **Privacy Policy**: Privacy policy published

Startup Validation

The application performs automatic startup checks:

```bash
# Run startup validation
python main.py

# Expected output:
✅ Python Version - Python 3.11.x
✅ Dependencies - All required packages installed
✅ Environment Variables - All variables configured
✅ Configuration Validity - All configuration values valid
✅ Database Connection - Connected to youtube_optimizer
✅ Redis Connection - Connected (version: 7.2.x)
✅ File Permissions - Write access to logs/ confirmed
✅ Anthropic API - Connected successfully
```

If any critical checks fail, the application will not start.

Environment Variables Validation

Required (Critical)

```bash
SECRET_KEY=<32+ character secure key>
DATABASE_PASSWORD=<strong password>
```

Required for AI Features

```bash
ANTHROPIC_API_KEY=sk-ant-<your-key>
SERPAPI_API_KEY=<your-key>
```

Required for YouTube Integration

```bash
YOUTUBE_API_KEY=<your-key>
YOUTUBE_CLIENT_ID=<your-client-id>
YOUTUBE_CLIENT_SECRET=<your-client-secret>
```

Optional but Recommended

```bash
SENTRY_DSN=<your-sentry-dsn>
REDIS_PASSWORD=<your-redis-password>
```

Performance Tuning

Database Connection Pool

```python
# Recommended settings for different loads

# Light load (< 100 concurrent users)
DATABASE_POOL_SIZE=10
DATABASE_MAX_OVERFLOW=20

# Medium load (100-500 concurrent users)
DATABASE_POOL_SIZE=20
DATABASE_MAX_OVERFLOW=40

# Heavy load (500+ concurrent users)
DATABASE_POOL_SIZE=50
DATABASE_MAX_OVERFLOW=100
```

Workers Configuration

```bash
```

```
# Formula: (2 x CPU cores) + 1
# 4 CPU cores = 9 workers
WORKERS=9

# Minimum for production: 4
WORKERS=4
```

Rate Limiting

```python
# Adjust based on your needs
DEFAULT_RATE_LIMIT="100/minute"  # Global default

# Endpoint-specific (in code)
channel_optimization: 10/minute
video_optimization: 30/minute
batch_operations: 5/hour
ai_requests: 5/minute
```

Monitoring Endpoints

Health Checks

```bash
# Basic health
GET /health

# Detailed health with dependencies
GET /health/detailed

# Readiness probe (K8s)
GET /health/ready

# Liveness probe (K8s)
GET /health/alive
```

Metrics

```bash
```

```
# Prometheus metrics
GET /metrics

# Circuit breaker status
GET /api/v1/system/circuit-breakers

# Application info
GET /info
```

Security Hardening

1. Firewall Rules

```bash
# Allow only necessary ports
ufw allow 80/tcp    # HTTP (for redirect to HTTPS)
ufw allow 443/tcp   # HTTPS
ufw allow 22/tcp    # SSH (from specific IPs only)
ufw deny incoming
ufw enable
```

2. Nginx Configuration

```nginx
```

```
# /etc/nginx/sites-available/youtube-optimizer

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

server {
    listen 443 ssl http2;
    server_name yourdomain.com;

    ssl_certificate /etc/letsencrypt/live/yourdomain.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/yourdomain.com/privkey.pem;

    # Security headers
    add_header X-Frame-Options "DENY";
    add_header X-Content-Type-Options "nosniff";
    add_header X-XSS-Protection "1; mode=block";
    add_header Strict-Transport-Security "max-age=31536000; includeSubDomains";

    # Proxy to FastAPI
    location / {
        proxy_pass http://127.0.0.1:8000;
        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;

        # Timeouts
        proxy_connect_timeout 60s;
        proxy_send_timeout 60s;
        proxy_read_timeout 60s;
    }

    # Rate limiting
    limit_req_zone $binary_remote_addr zone=api_limit:10m rate=10r/s;
    limit_req zone=api_limit burst=20 nodelay;
}
```

3. systemd Service

ini

```
# /etc/systemd/system/youtube-optimizer.service

[Unit]
Description=YouTube Channel Optimizer API
After=network.target postgresql.service redis.service

[Service]
Type=notify
User=appuser
Group=appuser
WorkingDirectory=/opt/youtube-optimizer
Environment="PATH=/opt/youtube-optimizer/venv/bin"
EnvironmentFile=/opt/youtube-optimizer/.env
ExecStart=/opt/youtube-optimizer/venv/bin/gunicorn main:app \
    --workers 4 \
    --worker-class uvicorn.workers.UvicornWorker \
    --bind 127.0.0.1:8000 \
    --timeout 300 \
    --graceful-timeout 30 \
    --keep-alive 5 \
    --max-requests 1000 \
    --max-requests-jitter 100
Restart=always
RestartSec=10

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

Deployment Commands

Docker Deployment

```bash
```

```bash
# Build
docker-compose build

# Start all services
docker-compose up -d

# Check logs
docker-compose logs -f api

# Check health
curl http://localhost:8000/health
```

Traditional Deployment

```bash
```

```bash
# Clone repository
git clone <your-repo> /opt/youtube-optimizer
cd /opt/youtube-optimizer

# Create virtual environment
python3.11 -m venv venv
source venv/bin/activate

# Install dependencies
pip install -r requirements.txt

# Configure environment
cp .env.example .env
nano .env  # Edit with your values

# Run database migrations
alembic upgrade head

# Test startup
python main.py

# Install as service
sudo cp youtube-optimizer.service /etc/systemd/system/
sudo systemctl daemon-reload
sudo systemctl enable youtube-optimizer
sudo systemctl start youtube-optimizer

# Check status
sudo systemctl status youtube-optimizer
```

Kubernetes Deployment

```bash
```

```bash
# Create namespace
kubectl create namespace youtube-optimizer

# Create secrets
kubectl create secret generic app-secrets \
  --from-env-file=.env \
  -n youtube-optimizer

# Deploy
kubectl apply -f k8s/ -n youtube-optimizer

# Check status
kubectl get pods -n youtube-optimizer
kubectl logs -f deployment/youtube-optimizer -n youtube-optimizer
```

Post-Deployment Verification

1. Smoke Tests

```bash
# Health check
curl https://yourdomain.com/health

# API info
curl https://yourdomain.com/info

# Metrics (if exposed)
curl https://yourdomain.com/metrics
```

2. Load Testing

```bash
# Using Apache Bench
ab -n 1000 -c 10 https://yourdomain.com/health

# Using wrk
wrk -t4 -c100 -d30s https://yourdomain.com/health
```

3. Monitor Logs

```bash
# Application logs
tail -f logs/app.log

# System logs
journalctl -u youtube-optimizer -f

# Docker logs
docker-compose logs -f api
```

Rollback Procedure

```bash
# Docker
docker-compose down
docker-compose pull  # Get previous version
docker-compose up -d

# Systemd
sudo systemctl stop youtube-optimizer
cd /opt/youtube-optimizer
git checkout <previous-commit>
pip install -r requirements.txt
sudo systemctl start youtube-optimizer

# Kubernetes
kubectl rollout undo deployment/youtube-optimizer -n youtube-optimizer
```

Emergency Contacts

- **On-Call Engineer**: [Contact Info]
- **Database Admin**: [Contact Info]
- **DevOps Team**: [Contact Info]
- **Security Team**: [Contact Info]

Support Resources

- **Documentation**: https://docs.yourdomain.com
- **Status Page**: https://status.yourdomain.com
- **Sentry**: https://sentry.io/your-org
- **Grafana**: https://grafana.yourdomain.com
- **Prometheus**: https://prometheus.yourdomain.com

✅ **ALL CHECKS COMPLETE - READY FOR PRODUCTION**