⚡ Production Quick Start Guide

1️⃣ Prerequisites

```bash
# System Requirements
- Python 3.11+
- PostgreSQL 14+
- Redis 7+
- 2GB+ RAM
- 10GB+ disk space

# Get API Keys
1. Anthropic: https://console.anthropic.com/
2. SerpAPI: https://serpapi.com/
3. YouTube: https://console.cloud.google.com/
```

2️⃣ Installation (5 minutes)

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

# Install dependencies
pip install -r requirements.txt

# Copy environment file
cp .env.example .env
```

3️⃣ Configuration (10 minutes)

Edit `.env` file:

```bash
```

```
# CRITICAL: Set these first
SECRET_KEY=$(python -c "import secrets; print(secrets.token_urlsafe(32))")
DATABASE_PASSWORD=your-strong-password-here

# AI/ML (Get from respective websites)
ANTHROPIC_API_KEY=sk-ant-your-key-here
SERPAPI_API_KEY=your-serpapi-key-here

# Database
DATABASE_HOST=localhost
DATABASE_PORT=5432
DATABASE_NAME=youtube_optimizer
DATABASE_USER=postgres

# Redis
REDIS_HOST=localhost
REDIS_PORT=6379

# Production settings
ENVIRONMENT=production
DEBUG=false
LOG_LEVEL=INFO
```

4️⃣ Database Setup (5 minutes)

```
bash

# Create database
createdb youtube_optimizer


# Run SQL schema
psql -d youtube_optimizer -f scripts/init-db.sql


# Or use migrations
alembic upgrade head
```

5️⃣ Validation (2 minutes)

```
bash
```

```
# Test configuration
python -c "from config import get_settings; print(get_settings().to_dict())"

# Run startup checks
python main.py

# Expected output:
# ✅ Python Version - Python 3.11.x
# ✅ Dependencies - All required packages installed
# ✅ Database Connection - Connected
# ✅ Anthropic API - Connected
# ✅ All critical startup checks passed
```

6️⃣ Run Application

Development:

```bash
python main.py
# or
uvicorn main:app --reload
```

Production:

```bash
gunicorn main:app \
  --workers 4 \
  --worker-class uvicorn.workers.UvicornWorker \
  --bind 0.0.0.0:8000 \
  --timeout 300
```

Docker:

```bash
docker-compose up -d
docker-compose logs -f api
```

7️⃣ Verify Deployment (1 minute)

```bash
# Health check
curl http://localhost:8000/health

# Expected response:
{
  "status": "healthy",
  "database": "connected",
  "redis": "connected"
}

# API info
curl http://localhost:8000/info

# API docs
open http://localhost:8000/docs
```

8️⃣ Test AI Features (2 minutes)

```bash
# Create test optimization
curl -X POST http://localhost:8000/api/v1/channel/1/optimize \
  -H "Authorization: Bearer <your-token>" \
  -H "Content-Type: application/json"

# Check status
curl http://localhost:8000/api/v1/channel/optimization/1/status
```

🔒 Security Checklist

Before going live:

- [ ] Change `SECRET_KEY` to secure random value
- [ ] Set strong `DATABASE_PASSWORD`
- [ ] Configure `ALLOWED_HOSTS` (remove `*`)
- [ ] Update `CORS_ORIGINS` with your domains

- □ Enable HTTPS/SSL
- □ Set `DEBUG=false`
- □ Configure `SENTRY_DSN` for error tracking

📊 Monitoring Setup

```bash
# Prometheus metrics
curl http://localhost:8000/metrics

# Circuit breaker status
curl http://localhost:8000/api/v1/system/circuit-breakers

# Health report
curl http://localhost:8000/health/detailed
```

🚨 Troubleshooting

Database connection fails:

```bash
# Check PostgreSQL is running
sudo systemctl status postgresql

# Test connection
psql -h localhost -U postgres -d youtube_optimizer
```

Redis connection fails:

```bash
# Check Redis is running
sudo systemctl status redis

# Test connection
redis-cli ping
```

Anthropic API fails:

```
bash

# Verify API key
echo $ANTHROPIC_API_KEY

# Test API
python -c "import anthropic; client = anthropic.Anthropic(api_key='$ANTHROPIC_API_KEY'); print('OK')"
```

Import errors:

```
bash

# Reinstall dependencies
pip install -r requirements.txt --force-reinstall
```

📈 Performance Tuning

For 100 concurrent users:

```
bash

WORKERS=4
DATABASE_POOL_SIZE=10
DATABASE_MAX_OVERFLOW=20
```

For 500+ concurrent users:

```
bash

WORKERS=8
DATABASE_POOL_SIZE=20
DATABASE_MAX_OVERFLOW=40
```

🔄 Updates & Maintenance

Update application:

```
bash
```

```
git pull
pip install -r requirements.txt
alembic upgrade head
sudo systemctl restart youtube-optimizer
```

Backup database:

```
bash

pg_dump youtube_optimizer > backup_$(date +%Y%m%d).sql
```

View logs:

```
bash

tail -f logs/app.log
# or
journalctl -u youtube-optimizer -f
```

📞 Support

- **Documentation**: `/docs` endpoint
- **Health Status**: `/health` endpoint
- **Metrics**: `/metrics` endpoint
- **Logs**: `logs/app.log`

✅ Production Checklist

□ All environment variables set

□ Database initialized and accessible

□ Redis running and accessible

□ API keys configured and tested

□ Startup validation passes

□ Health check returns healthy

□ HTTPS configured

□ Firewall rules set

□ Monitoring configured

- [ ] Backups automated
- [ ] Documentation reviewed

🎉 You're ready for production!

Total setup time: ~30 minutes

Your application is now enterprise-ready with:

- ✅ AI-powered optimization
- ✅ Production database management
- ✅ Circuit breakers
- ✅ Comprehensive monitoring
- ✅ Security hardening
- ✅ Auto-scaling ready