🚀 Deployment Guide - Production Ready

Complete guide for deploying Invoice Generator API to production.

📋 Pre-Deployment Checklist

- ☐ Change `SECRET_KEY` to a strong random string
- ☐ Set `DEBUG=false` in production
- ☐ Configure email service (SendGrid recommended)
- ☐ Set up PostgreSQL database (or keep SQLite for small scale)
- ☐ Configure HTTPS/SSL certificate
- ☐ Set up domain name
- ☐ Configure CORS allowed origins
- ☐ Set up monitoring (optional)
- ☐ Configure backups
- ☐ Test all endpoints

🌐 Deployment Options

Option 1: Railway (Easiest - Recommended for MVP)

Free Tier: 500 hours/month, perfect for MVP!

Steps:

1. **Create Railway Account**
   - Go to https://railway.app
   - Sign up with GitHub
2. **Deploy from GitHub**

```bash
# Push your code to GitHub first
git init
git add .
git commit -m "Initial commit"
git remote add origin https://github.com/yourusername/invoice-api.git
git push -u origin main
```

3. **Connect to Railway**
   - Click "New Project"
   - Select "Deploy from GitHub repo"
   - Choose your repository
   - Railway will auto-detect FastAPI
4. **Add Environment Variables**
   - Go to project settings → Variables
   - Add all variables from `.env`:

```
SECRET_KEY=your-production-secret-key-here
DATABASE_URL=postgresql://...
EMAIL_HOST=smtp.sendgrid.net
EMAIL_PORT=587
EMAIL_USERNAME=apikey
EMAIL_PASSWORD=your-sendgrid-key
EMAIL_FROM=noreply@yourdomain.com
DEBUG=false
ALLOWED_ORIGINS=https://yourdomain.com
```

5. **Add PostgreSQL Database**
   - Click "New" → "Database" → "PostgreSQL"
   - Railway will automatically set `DATABASE_URL`
6. **Deploy!**
   - Railway will automatically build and deploy
   - Your API will be live at: `https://your-app.railway.app`

Cost: Free for 500 hours/month, then $5-20/month

Option 2: Render.com (Great Alternative)

Free Tier: 750 hours/month

Steps:

1. **Create Render Account**
   - Go to https://render.com
   - Sign up
2. **Create Web Service**
   - Click "New +" → "Web Service"
   - Connect GitHub repo
   - Configure:
     - Name: `invoice-api`
     - Environment: `Python 3`
     - Build Command: `pip install -r requirements.txt`
     - Start Command: `uvicorn app.main:app --host 0.0.0.0 --port $PORT`
3. **Add PostgreSQL Database**
   - Click "New +" → "PostgreSQL"
   - Connect to your web service
4. **Environment Variables**
   - Add same variables as Railway
5. **Deploy**
   - Click "Create Web Service"

Cost: Free tier available, paid starts at $7/month

Option 3: Docker + VPS (Most Control)

Perfect for **DigitalOcean, Linode, AWS EC2, Azure**.

Prerequisites:

- VPS with Ubuntu 22.04
- Domain name pointed to VPS IP
- SSH access

Setup Steps:

1. **Connect to VPS**

```bash
ssh root@your-vps-ip
```

2. **Install Docker**

```bash
# Update packages
apt update && apt upgrade -y

# Install Docker
curl -fsSL https://get.docker.com -o get-docker.sh
sh get-docker.sh

# Install Docker Compose
apt install docker-compose -y
```

3. **Clone Repository**

```bash
cd /opt
git clone https://github.com/yourusername/invoice-api.git
cd invoice-api
```

4. **Configure Environment**

```bash
cp .env.example .env
nano .env
# Edit with your production values
```

5. **Start Services**

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

6. **Setup Nginx Reverse Proxy**

```bash
apt install nginx certbot python3-certbot-nginx -y
```

Create Nginx config:

```bash
nano /etc/nginx/sites-available/invoice-api
```

Add:

```nginx
server {
    listen 80;
    server_name api.yourdomain.com;

    location / {
        proxy_pass http://localhost: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;
    }
}
```

Enable site:

```bash
ln -s /etc/nginx/sites-available/invoice-api /etc/nginx/sites-enabled/
nginx -t
systemctl restart nginx
```

7. **Get SSL Certificate**

```bash
certbot --nginx -d api.yourdomain.com
```

8. **Setup Auto-Restart**

```bash
# Docker will auto-restart on failure
docker update --restart=unless-stopped invoice_api
```

Cost: $5-20/month depending on VPS provider

Option 4: Heroku (Simple but Expensive)

Steps:

1. **Install Heroku CLI**

```bash
curl https://cli-assets.heroku.com/install.sh | sh
```

2. **Login**

```bash
heroku login
```

3. **Create App**

```bash
heroku create invoice-api-prod
```

4. **Add PostgreSQL**

```bash
heroku addons:create heroku-postgresql:mini
```

5. **Set Environment Variables**

```bash
heroku config:set SECRET_KEY=your-secret-key
heroku config:set EMAIL_HOST=smtp.sendgrid.net
# ... add all variables
```

6. **Create Procfile**

```bash
echo "web: uvicorn app.main:app --host 0.0.0.0 --port \$PORT" > Procfile
```

7. **Deploy**

```bash
git push heroku main
```

Cost: Starts at $7/month (no free tier anymore)

🔒 Security Best Practices

1. Strong SECRET_KEY

```bash
# Generate secure key
python -c "import secrets; print(secrets.token_urlsafe(64))"
```

2. HTTPS Only

```python
# In app/main.py, add:
from fastapi.middleware.httpsredirect import HTTPSRedirectMiddleware
app.add_middleware(HTTPSRedirectMiddleware)
```

3. Rate Limiting (Production)

```bash
pip install slowapi

# In app/main.py:
from slowapi import Limiter, _rate_limit_exceeded_handler
from slowapi.util import get_remote_address

limiter = Limiter(key_func=get_remote_address)
app.state.limiter = limiter
app.add_exception_handler(429, _rate_limit_exceeded_handler)
```

4. CORS Configuration

```python
# Only allow your domains
ALLOWED_ORIGINS=https://yourdomain.com,https://app.yourdomain.com
```

5. Database Backups

```bash
# Automated backup script (run daily via cron)
#!/bin/bash
DATE=$(date +%Y%m%d)
docker exec invoice_db pg_dump -U invoice_user invoice_db > backup_$DATE.sql
# Upload to S3 or backup service
```

📊 Monitoring & Logging

Option 1: Sentry (Error Tracking)

```bash
pip install sentry-sdk[fastapi]
```

```python
# In app/main.py:
import sentry_sdk

sentry_sdk.init(
    dsn="your-sentry-dsn",
    traces_sample_rate=1.0,
)
```

Option 2: Logfire (FastAPI Native)

```bash
pip install logfire
```

```python
import logfire
logfire.configure()
logfire.instrument_fastapi(app)
```

Option 3: Simple File Logging

```python
# In app/main.py:
import logging

logging.basicConfig(
    filename='app.log',
    level=logging.INFO,
    format='%(asctime)s - %(name)s - %(levelname)s - %(message)s'
)
```

🔄 CI/CD Pipeline (GitHub Actions)

Create `.github/workflows/deploy.yml`:

```yaml
name: Deploy to Production

on:
  push:
    branches: [ main ]

jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - name: Set up Python
        uses: actions/setup-python@v4
        with:
          python-version: '3.11'
      - name: Install dependencies
        run: |
          pip install -r requirements.txt
          pip install pytest
      - name: Run tests
        run: pytest

  deploy:
    needs: test
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - name: Deploy to Railway
        run: |
          curl -fsSL https://railway.app/install.sh | sh
          railway up
        env:
          RAILWAY_TOKEN: ${{ secrets.RAILWAY_TOKEN }}
```

📦 Database Migration

Using Alembic

1. **Initialize Alembic**

```bash
alembic init alembic
```

2. **Configure** Edit `alembic/env.py`:

```python
from app.database import Base
from app.models import user, invoice

target_metadata = Base.metadata
```

3. **Create Migration**

```bash
alembic revision --autogenerate -m "Initial migration"
```

4. **Apply Migration**

```bash
alembic upgrade head
```

5. **In Production**

```bash
# Add to Procfile or startup script
alembic upgrade head && uvicorn app.main:app
```

🌍 Publishing to RapidAPI

Steps:

1. **Deploy Your API** (use any option above)
2. **Get Production URL** (e.g., `https://invoice-api.railway.app`)
3. **Create RapidAPI Account**
   - Go to https://rapidapi.com/provider
   - Sign up as provider
4. **Add New API**
   - Click "Add New API"
   - Fill details:
     - Name: Arabic Invoice Generator API
     - Category: Business
     - Base URL: Your production URL
     - Visibility: Public/Private
5. **Configure Endpoints**
   - Import OpenAPI spec from `/openapi.json`
   - RapidAPI will auto-detect all endpoints
6. **Set Pricing Tiers**

```
Free: 10 invoices/month - $0
Basic: 100 invoices/month - $9.99
Pro: 500 invoices/month - $29.99
Ultra: Unlimited - $99.99
```

7. **Test & Publish**
   - Test all endpoints
   - Submit for review
   - Go live! 🚀

🧪 Load Testing

```bash
# Install locust
pip install locust

# Create locustfile.py
```

```python
from locust import HttpUser, task, between

class InvoiceUser(HttpUser):
    wait_time = between(1, 3)

    def on_start(self):
        # Login
        response = self.client.post("/auth/login", json={
            "username": "test",
            "password": "test"
        })
        self.token = response.json()["access_token"]

    @task
    def create_invoice(self):
        self.client.post(
            "/invoices/generate",
            headers={"Authorization": f"Bearer {self.token}"},
            json={
                "client_name": "Test Client",
                "client_email": "test@test.com",
                "language": "en",
                "currency": "USD",
                "items": [{"name": "Service", "quantity": 1, "price": 100}]
            }
        )
```

```bash
# Run load test
locust -f locustfile.py --host=https://your-api.com
```

📈 Performance Optimization

1. Add Redis Caching (Optional)

```bash
pip install redis aioredis
```

```python
from fastapi_cache import FastAPICache
from fastapi_cache.backends.redis import RedisBackend

@app.on_event("startup")
async def startup():
    redis = aioredis.from_url("redis://localhost")
    FastAPICache.init(RedisBackend(redis), prefix="invoice-cache")
```

2. Database Connection Pooling

```python
# In database.py
engine = create_engine(
    settings.DATABASE_URL,
    pool_size=20,
    max_overflow=0,
    pool_pre_ping=True
)
```

3. Increase Workers

```bash
# Instead of single worker
uvicorn app.main:app --workers 4
```

🆘 Troubleshooting Production Issues

Issue: App crashes on startup

```bash
# Check logs
docker logs invoice_api
# or
heroku logs --tail
```

Issue: Database connection errors

```bash
# Test connection
python -c "from app.database import engine; engine.connect()"
```

Issue: Email not sending

```bash
# Test SMTP
python -c "import smtplib; smtplib.SMTP('smtp.sendgrid.net', 587).starttls()"
```

Issue: High memory usage

```bash
# Monitor resources
docker stats
# Optimize PDF generation
# Consider background tasks for heavy operations
```

✅ Post-Deployment Checklist

- [ ] Test all endpoints in production
- [ ] Verify email sending works
- [ ] Test PDF generation
- [ ] Check HTTPS certificate
- [ ] Set up monitoring alerts
- [ ] Configure database backups
- [ ] Update documentation with prod URL
- [ ] Test rate limiting
- [ ] Monitor error logs
- [ ] Set up status page (optional)

🎉 You're Live!

Your Invoice Generator API is now running in production! 🚀

Next Steps:

1. Share your API on RapidAPI
2. Create marketing materials
3. Get your first users
4. Collect feedback
5. Iterate and improve

Support:

- Monitor Sentry for errors
- Check Railway/Render dashboard
- Review logs daily
- Update dependencies monthly

Production Deployment Complete! Happy invoicing! 💼