🐳 Docker Deployment Guide for ORBAPI OCR Service

📋 Prerequisites

- Docker Engine 20.10+
- Docker Compose 2.0+
- At least 8GB RAM
- 4 CPU cores recommended

🚀 Quick Start

1. Build and Run with Docker Compose (Recommended)

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

# View logs
docker-compose logs -f ocr-api

# Check status
docker-compose ps

# Stop the service
docker-compose down
```

2. Build and Run with Docker

```
# Build the image
docker build -t orbapi-ocr:latest .

# Run the container
docker run -d \
  --name orbapi-ocr \
  -p 8000:8000 \
  -v $(pwd)/models:/app/models:ro \
  -v $(pwd)/logs:/app/logs \
  orbapi-ocr:latest

# View logs
docker logs -f orbapi-ocr

# Stop the container
```

```
docker stop orbapi-ocr
docker rm orbapi-ocr
```

Directory Structure

```
ORBAPI/
├── Dockerfile             # Docker image definition
├── docker-compose.yml     # Docker Compose orchestration
├── .dockerignore         # Files to exclude from build
├── fastapi_server_new.py # Main FastAPI application
├── models/               # YOLO models (mounted as volume)
│   └── pt/
├── lockup/               # Template images for ORB alignment
├── weights/              # OCR model weights
├── logs/                 # Application logs (persistent)
│   └── tasks/
└── images/               # Uploaded images (optional)
```

Configuration

Environment Variables

Edit `docker-compose.yml` to customize:

```
environment:
  - PYTHONUNBUFFERED=1        # Real-time log output
  - NNPACK_WARN=0             # Suppress NNPACK warnings
  - TZ=Asia/Ho_Chi_Minh       # Timezone
  # Add your custom variables:
  # - API_KEY=your_secret_key
  # - MAX_WORKERS=4
```

Resource Limits

Adjust CPU and memory limits in `docker-compose.yml`:

```
deploy:
  resources:
    limits:
      cpus: '4'           # Maximum 4 CPU cores
      memory: 8G          # Maximum 8GB RAM
    reservations:
      cpus: '2'           # Minimum 2 CPU cores
      memory: 4G          # Minimum 4GB RAM
```

Port Configuration

Change the exposed port:

```
ports:
  - "8080:8000"  # External:Internal
```

🔍 Health Check

The service includes built-in health checks:

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

# Check Docker health status
docker inspect --format='{{.State.Health.Status}}' orbapi-ocr
```

Expected response:

```
{
  "status": "healthy",
  "engines": {
    "paddleocr": {"available": true, "status": "ready"},
    "vietocr": {"available": true, "status": "ready"}
  }
}
```

📦 Volumes

Persistent Data

Volumes mounted by default:

1. **Models** (read-only): `./models:/app/models:ro`
2. **Lockup templates** (read-only): `./lockup:/app/lockup:ro`
3. **Weights** (read-only): `./weights:/app/weights:ro`
4. **Logs** (read-write): `./logs:/app/logs`
5. **Images** (read-write): `./images:/app/images`

Update Models Without Rebuild

```
# Copy new model to host
cp new_model.pt ./models/pt/

# Restart container to reload
docker-compose restart ocr-api
```

🌐 API Access

Once running, access:

- **Web Interface**: http://localhost:8000
- **API Documentation**: http://localhost:8000/docs
- **Health Check**: http://localhost:8000/health
- **Scan API**: http://localhost:8000/api/scan/

Test with curl

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

# Scan image
curl -X POST "http://localhost:8000/api/scan/" \
  -H "Content-Type: multipart/form-data" \
  -F "file=@test_image.jpg"
```

🔧 Troubleshooting

Container won't start

```
# Check logs
docker-compose logs ocr-api

# Common issues:
# 1. Port 8000 already in use
docker-compose down
lsof -i :8000  # Find process using port
kill -9 <PID>

# 2. Insufficient memory
# Increase Docker Desktop memory allocation to at least 8GB
```

Models not loading

```
# Verify models directory
ls -la ./models/pt/

# Check volume mounts
docker inspect orbapi-ocr | grep Mounts -A 20

# Rebuild if needed
docker-compose down
docker-compose up -d --build
```

Slow performance

```
# Increase workers in Dockerfile
CMD ["uvicorn", "fastapi_server_new:app", "--host", "0.0.0.0", "--port",
"8000", "--workers", "8"]

# Or increase CPU/memory limits in docker-compose.yml
```

Out of memory errors

```
# Reduce workers
CMD ["uvicorn", "fastapi_server_new:app", "--host", "0.0.0.0", "--port",
"8000", "--workers", "2"]

# Increase memory limit
docker-compose down
# Edit docker-compose.yml: memory: 12G
docker-compose up -d
```

📊 Monitoring

View real-time logs

```
# All logs
docker-compose logs -f ocr-api

# Last 100 lines
docker-compose logs --tail=100 ocr-api

# Filter for errors
docker-compose logs ocr-api | grep ERROR
```

Resource usage

```
# CPU and memory usage
docker stats orbapi-ocr

# Disk usage
docker system df
```

Access container shell

```
# Interactive bash
docker exec -it orbapi-ocr bash

# Check processes
docker exec orbapi-ocr ps aux

# Check Python version
docker exec orbapi-ocr python --version
```

🔄 Updates and Maintenance

Update application code

```
# Pull latest code
git pull origin main

# Rebuild and restart
docker-compose down
docker-compose up -d --build
```

Backup

```
# Backup models
tar -czf models_backup_$(date +%Y%m%d).tar.gz models/

# Backup logs
tar -czf logs_backup_$(date +%Y%m%d).tar.gz logs/

# Export Docker image
docker save orbapi-ocr:latest | gzip > orbapi-ocr_latest.tar.gz
```

Restore

```
# Restore models
tar -xzf models_backup_20251004.tar.gz

# Load Docker image
docker load < orbapi-ocr_latest.tar.gz
```

Clean up

```
# Remove stopped containers
docker container prune

# Remove unused images
docker image prune -a

# Remove unused volumes
docker volume prune

# Full cleanup (CAUTION: removes all unused Docker resources)
docker system prune -a --volumes
```

🚀 Production Deployment

Use production server

Edit `Dockerfile`:

```
# Change from development server
CMD ["gunicorn", "fastapi_server_new:app", \
     "--workers", "4", \
     "--worker-class", "uvicorn.workers.UvicornWorker", \
     "--bind", "0.0.0.0:8000", \
     "--timeout", "300"]
```

Add gunicorn to requirements:

```
echo "gunicorn" >> requirements.txt
```

Enable HTTPS

Use a reverse proxy like Nginx:

```yaml
# docker-compose.yml
services:
  nginx:
    image: nginx:alpine
    ports:
      - "443:443"
      - "80:80"
    volumes:
      - ./nginx.conf:/etc/nginx/nginx.conf:ro
      - ./ssl:/etc/nginx/ssl:ro
    depends_on:
      - ocr-api
```

Auto-restart on failure

Already configured in `docker-compose.yml`:

```yaml
restart: unless-stopped
```

📝 Notes

1. **First build takes time**: Downloading dependencies (~5-10 minutes)
2. **Model loading**: Service needs 30-60 seconds to load all models
3. **Memory usage**: Expect 2-4GB base + 500MB per concurrent request
4. **GPU support**: Not included. Add NVIDIA Docker runtime for GPU acceleration
5. **Scaling**: Use Kubernetes or Docker Swarm for multi-node deployment

---

🆘 Support

If you encounter issues:

1. Check logs: `docker-compose logs ocr-api`
2. Verify health: `curl http://localhost:8000/health`
3. Check resources: `docker stats`
4. Review this guide's Troubleshooting section

📚 Additional Resources

- FastAPI Documentation
- Docker Compose Reference
- Uvicorn Deployment

Last updated: October 4, 2025