**Step 7C: Docker Production Deployment - Detailed Documentation**

**Overview**

This document chronicles the complete implementation of Step 7C: Production Docker Deployment for the NBE Prediction API, including all problems encountered and their solutions.

**Initial Setup and File Creation**

**Files Created Successfully ✅**

1. **Dockerfile.production** - Multi-stage production Docker configuration
2. **docker-compose.yml** - Complete service orchestration
3. **nginx.conf** - Reverse proxy configuration
4. **.env.production** - Production environment variables
5. **health\_check.py** - Container health monitoring
6. **requirements.production.txt** - Production dependencies
7. **deploy.sh** - Automated deployment script

**Project Structure Achieved**

reha\_assist\_iru/

├── code/step7\_deployment/

│ ├── Dockerfile.production

│ ├── Dockerfile.simple

│ ├── Dockerfile.working

│ ├── nginx.conf

│ ├── .env.production

│ ├── health\_check.py

│ ├── requirements.production.txt

│ └── deploy.sh

├── docker-compose.yml

├── models/ (with trained ML models)

└── ... (existing project files)

**Problem 1: Gunicorn Module Not Found ❌**

**Issue Description**

* **Error**: ModuleNotFoundError: No module named 'gunicorn'
* **Root Cause**: The multi-stage Dockerfile was looking for requirements.txt in the wrong location
* **Impact**: Container kept restarting, unable to start the API

**First Attempted Solution (Failed)**

* Modified Dockerfile path from requirements.txt to code/step7\_deployment/requirements.production.txt
* **Result**: Still failed because the requirements file content wasn't being properly used

**Investigation Findings**

* Docker build process was installing packages correctly (confirmed by build logs)
* Packages were being installed in /root/.local/bin during build stage
* Runtime user apiuser couldn't access packages due to PATH issues

**Problem 2: PATH Configuration Issues ❌**

**Issue Description**

* **Error**: Still No module named 'gunicorn' despite successful package installation
* **Root Cause**: Multi-stage Docker build with user switching caused PATH misalignment
* **Analysis**: Packages installed as root in builder stage, but runtime used non-root user

**Attempted Solutions (All Failed)**

1. **PATH Environment Variable Fix**:
2. ENV PATH=/home/apiuser/.local/bin:/root/.local/bin:$PATH

**Result**: Still couldn't find packages

1. **Python Module Command**:
2. CMD ["python", "-m", "gunicorn", ...]

**Result**: Python couldn't import gunicorn module

1. **Complex Multi-stage Copy**:
2. COPY --from=builder /root/.local /home/apiuser/.local

**Result**: Path issues persisted

**Problem 3: Package Installation Architecture Issues ❌**

**Issue Description**

* **Error**: gcc failed: No such file or directory during package compilation
* **Root Cause**: Missing build tools for compiling Python packages like scikit-learn and xgboost
* **Impact**: Docker build process failing during pip install phase

**Solution Attempts**

1. **Added Build Tools**: Added gcc, g++ to Dockerfile
2. **Simplified Architecture**: Moved from multi-stage to single-stage build
3. **Python Version Issues**: Discovered Python 3.13 compatibility problems

**Problem 4: Python 3.13 Compatibility Issues ❌**

**Issue Description**

* **Error**:
* Cython.Compiler.Errors.CompileError: sklearn/linear\_model/\_cd\_fast.pyxerror: metadata-generation-failed
* **Root Cause**: Python 3.13 is too new for stable scikit-learn compilation
* **Impact**: Unable to install machine learning packages

**Solution Applied ✅**

* **Downgraded to Python 3.11**: Changed base image from python:3.13-slim to python:3.11-slim
* **Rationale**: Python 3.11 has pre-compiled wheel packages for ML libraries
* **Result**: Package installation succeeded

**Problem 5: NumPy Compatibility Issues ❌ (Current Issue)**

**Issue Description**

* **Error**: No module named 'numpy.\_core'
* **Root Cause**: NumPy version incompatibility between training environment and container
* **Impact**: API starts but fails during model loading phase
* **Status**: Currently unresolved

**Current Situation Analysis**

1. **Container Startup**: ✅ Successfully starts
2. **Package Installation**: ✅ All packages install correctly
3. **API Framework**: ✅ FastAPI/Uvicorn starts properly
4. **Model Loading**: ❌ Fails due to numpy.\_core module missing

**Attempted Solutions (Current)**

1. **NumPy Version Downgrade**: Changed from numpy==1.25.2 to numpy==1.24.3
2. **Complete Cache Clearing**: Multiple attempts with docker system prune -a -f
3. **Fresh Rebuilds**: Used --no-cache --pull flags
4. **Result**: Still experiencing the same numpy.\_core error

**Technical Stack Evolution**

**Current Working Configuration**

* **Base Image**: python:3.11-slim
* **Architecture**: Single-stage Dockerfile (simplified from multi-stage)
* **Package Management**: Global pip installation (removed --user flag)
* **Server**: Uvicorn (simplified from Gunicorn)
* **User Security**: Non-root user apiuser

**Current Dockerfile.working**

FROM python:3.11-slim

# Create non-root user

RUN groupadd -r apiuser && useradd -r -g apiuser -d /app -s /bin/bash apiuser

# Set working directory

WORKDIR /app

# Install system dependencies

RUN apt-get update && apt-get install -y \

curl \

&& rm -rf /var/lib/apt/lists/\*

# Copy requirements and install packages globally

COPY code/step7\_deployment/requirements.production.txt requirements.txt

RUN pip install --no-cache-dir -r requirements.txt

# Copy application code

COPY . .

# Create necessary directories and set permissions

RUN mkdir -p /app/logs /app/data /app/models /app/plots \

&& chown -R apiuser:apiuser /app

# Copy production configuration

COPY code/step7\_deployment/.env.production /app/.env

# Set environment

ENV PYTHONPATH=/app

# Switch to non-root user

USER apiuser

# Expose port

EXPOSE 8000

# Start with uvicorn

CMD ["uvicorn", "code.step6\_api\_development.api\_main:app", "--host", "0.0.0.0", "--port", "8000", "--workers", "1"]

**Docker Compose Configuration**

version: '3.8'

services:

nbe-api:

build:

context: .

dockerfile: code/step7\_deployment/Dockerfile.working

image: nbe-prediction-api:1.0.0

container\_name: nbe-prediction-api

restart: unless-stopped

environment:

- ENVIRONMENT=production

- REQUIRE\_AUTH=true

- ENABLE\_RATE\_LIMITING=true

- LOG\_LEVEL=INFO

- CORS\_ORIGINS=http://enaio-server,https://enaio-server

- PROJECT\_ROOT=/app

volumes:

- ./logs:/app/logs

- ./models:/app/models:ro

- api-data:/app/data

ports:

- "8000:8000"

networks:

- nbe-network

volumes:

api-data:

driver: local

networks:

nbe-network:

driver: bridge

**Current Requirements.production.txt**

fastapi==0.104.1

uvicorn[standard]==0.24.0

gunicorn==21.2.0

scikit-learn==1.3.2

xgboost==2.0.2

pandas==2.1.4

numpy==1.24.3 # Fixed version for Python 3.11 compatibility

pydantic==2.5.0

python-multipart==0.0.6

python-jose[cryptography]==3.3.0

passlib[bcrypt]==1.7.4

slowapi==0.1.9

python-dotenv==1.0.0

requests==2.31.0

psutil==5.9.6

uvloop==0.19.0

httptools==0.6.1

python-dateutil==2.8.2

**Progress Summary**

**Successful Achievements ✅**

1. **Docker Environment**: Successfully configured Docker Desktop on Windows
2. **File Structure**: Created complete production deployment file structure
3. **Build Process**: Resolved compilation issues with Python version downgrade
4. **Container Startup**: API container starts and framework initializes
5. **Package Installation**: All required packages install successfully
6. **Service Architecture**: Uvicorn server starts and accepts connections

**Current Blocker ❌**

* **NumPy Module Loading**: No module named 'numpy.\_core' during model loading
* **Impact**: API starts but cannot load trained ML models
* **Criticality**: High - prevents functional API deployment

**Next Steps Required**

**Immediate Actions Needed**

1. **Root Cause Analysis**: Investigate numpy.\_core module structure in container
2. **Environment Comparison**: Compare working development environment with container environment
3. **Package Compatibility**: Verify numpy/scikit-learn/xgboost version compatibility matrix
4. **Alternative Approaches**: Consider using development requirements.txt as fallback

**Potential Solutions to Investigate**

1. **Use Development Requirements**: Replace production requirements with working development requirements.txt
2. **NumPy Version Matrix**: Test different numpy versions (1.23.x, 1.22.x series)
3. **Container Debugging**: Exec into container to manually test package imports
4. **Base Image Change**: Consider using different Python 3.11 base image variant

**Business Impact**

* **Current Status**: Production deployment blocked by numpy compatibility
* **Development API**: Still functional and ready for enaio integration
* **Risk Level**: Medium - can proceed with development environment temporarily
* **Timeline Impact**: Estimated 1-2 hours additional debugging needed

**This comprehensive documentation provides the complete context for troubleshooting the remaining numpy.\_core module issue.**

**Please provide the current error message you're seeing, and we can proceed with the most appropriate solution approach.**