🪟 Windows Server 2019 Deployment Guide

Prerequisites Check

1. Verify Python Installation

```cmd
python --version
```

Should show Python 3.9 or higher

2. Verify Node.js and npm

```cmd
node --version
npm --version
```

Should show Node.js 18+ and npm 9+

3. Verify PostgreSQL

```cmd
psql --version
```

Step 1: Update Environment Files

1.1 Backend .env

Navigate to backend folder and update .env:

```cmd
cd C:\Users\slg\Sea_Level_Dashboard_AWS_Ver_20_8_25\backend
notepad .env
```

Use the fixed backend .env content provided earlier.

1.2 Frontend .env

```cmd
cd ..\frontend
notepad .env
```

Use the fixed frontend .env content provided earlier.

Step 2: Build Frontend (Windows Commands)

2.1 Navigate to Frontend Directory

```cmd
cd C:\Users\slg\Sea_Level_Dashboard_AWS_Ver_20_8_25\frontend
```

2.2 Clean Previous Installation (Windows)

```cmd
:: Remove node_modules directory
rmdir /s /q node_modules

:: Remove package-lock.json
del package-lock.json

:: Or use PowerShell
powershell -Command "Remove-Item -Path node_modules -Recurse -Force -ErrorAction SilentlyContinue"
powershell -Command "Remove-Item -Path package-lock.json -Force -ErrorAction SilentlyContinue"
```

2.3 Install Dependencies

```cmd
npm install
```

If you get errors, try:

```cmd
npm install --force
:: or
npm install --legacy-peer-deps
```

2.4 Build for Production

```cmd
npm run build
```

This creates a `build` folder with optimized files.

2.5 Verify Build

```cmd
dir build
```

Should show files including `index.html`

Step 3: Setup Backend Server

3.1 Navigate to Backend

```cmd
cd C:\Users\slg\Sea_Level_Dashboard_AWS_Ver_20_8_25\backend
```

3.2 Save the local_server.py

Create or replace `local_server.py` with the Windows-compatible version provided above:

```cmd
notepad local_server.py
```

Paste the content and save.

3.3 Activate Python Virtual Environment

```cmd
:: If you have a venv
venv\Scripts\activate

:: Or create one if needed
python -m venv venv
venv\Scripts\activate
```

3.4 Install/Update Python Dependencies

```cmd
pip install fastapi uvicorn sqlalchemy psycopg2-binary python-dotenv pandas
```

Step 4: Configure Windows Firewall

4.1 Open Windows Firewall Settings

```cmd
```

```
:: Run as Administrator
netsh advfirewall firewall add rule name="Sea Level Dashboard" dir=in action=allow protocol=TCP localport=3088
netsh advfirewall firewall add rule name="Sea Level Dashboard UDP" dir=in action=allow protocol=UDP localport=
```

4.2 Verify Firewall Rules

```cmd
netsh advfirewall firewall show rule name="Sea Level Dashboard"
```

Step 5: Run the Server

5.1 Test Run

```cmd
cd C:\Users\slg\Sea_Level_Dashboard_AWS_Ver_20_8_25\backend
python local_server.py
```

5.2 Check if Server is Running

Open a new command prompt:

```cmd
:: Check if port is listening
netstat -an | findstr :30886

:: Test health endpoint
curl http://127.0.0.1:30886/health
```

Or open browser and visit:

- http://127.0.0.1:30886
- http://127.0.0.1:30886/docs

5.3 Test External Access

From another computer or phone on the network:

- http://5.102.231.16:30886
- http://5.102.231.16:30886/docs

Step 6: Run as Windows Service (Optional)

6.1 Install NSSM (Non-Sucking Service Manager)

1. Download NSSM from: https://nssm.cc/download
2. Extract to `C:\nssm\`

6.2 Create Service

Run as Administrator:

```cmd
cd C:\nssm\win64
nssm install SeaLevelDashboard

:: In the GUI that opens:
:: Path: C:\Users\slg\Sea_Level_Dashboard_AWS_Ver_20_8_25\backend\venv\Scripts\python.exe
:: Startup directory: C:\Users\slg\Sea_Level_Dashboard_AWS_Ver_20_8_25\backend
:: Arguments: local_server.py
```

6.3 Start Service

```cmd
nssm start SeaLevelDashboard
```

6.4 Check Service Status

```cmd
nssm status SeaLevelDashboard
```

Step 7: Alternative - Use Task Scheduler

7.1 Create Batch File

Create `start_server.bat`:

```batch
@echo off
cd /d C:\Users\slg\Sea_Level_Dashboard_AWS_Ver_20_8_25\backend
call venv\Scripts\activate
python local_server.py
```

7.2 Create Task

6. Program: `C:\Users\slg\Sea_Level_Dashboard_AWS_Ver_20_8_25\backend\start_server.bat`
7. Check "Run with highest privileges"

Troubleshooting Windows-Specific Issues

Issue: 'rm' is not recognized

Solution: Use Windows commands:

```cmd
:: Instead of: rm -rf node_modules
rmdir /s /q node_modules

:: Instead of: rm file.txt
del file.txt
```

Issue: Port 30886 Already in Use

Solution: Find and kill the process:

```cmd
:: Find process using port
netstat -ano | findstr :30886

:: Note the PID (last column), then kill it
taskkill /PID <PID> /F
```

Issue: PostgreSQL Connection Failed

Solution: Check PostgreSQL service:

```cmd
```

```
:: Check if PostgreSQL is running
sc query postgresql-x64-13

:: Start if stopped
net start postgresql-x64-13
```

Issue: npm Command Not Found

Solution: Add npm to PATH:

```cmd
:: Check npm location
where npm

:: Add to PATH if not found
setx PATH "%PATH%;C:\Program Files\nodejs"

:: Restart command prompt after this
```

Issue: Python Module Not Found

Solution: Ensure virtual environment is activated:

```cmd
:: Check if in venv
where python

:: Should show path like:
:: C:\...\backend\venv\Scripts\python.exe

:: If not, activate venv:
cd C:\Users\slg\Sea_Level_Dashboard_AWS_Ver_20_8_25\backend
venv\Scripts\activate
```

Performance Monitoring

Check Server Logs

```cmd
cd C:\Users\slg\Sea_Level_Dashboard_AWS_Ver_20_8_25\backend\logs
type server.log
```

Monitor in Real-Time

```cmd
:: Use PowerShell
powershell -Command "Get-Content server.log -Wait -Tail 50"
```

Check Memory Usage

```cmd
tasklist /FI "IMAGENAME eq python.exe"
```

Security Considerations for Windows Server

1. Windows Defender Exclusions

Add exclusions for better performance:

```powershell
# Run PowerShell as Administrator
Add-MpPreference -ExclusionPath "C:\Users\slg\Sea_Level_Dashboard_AWS_Ver_20_8_25"
Add-MpPreference -ExclusionProcess "python.exe"
Add-MpPreference -ExclusionProcess "node.exe"
```

2. IIS Reverse Proxy (Optional)

If you have IIS installed, you can set up a reverse proxy:

1. Install URL Rewrite and Application Request Routing
2. Create a new site in IIS
3. Add reverse proxy rules to forward to localhost:30886

Verification Checklist

Run these commands to verify everything is working:

```cmd
```

```
:: 1. Check Python
python --version

:: 2. Check Node
node --version

:: 3. Check if server is running
netstat -an | findstr :30886

:: 4. Test local access
curl http://127.0.0.1:30886/health

:: 5. Check frontend build exists
dir C:\Users\slg\Sea_Level_Dashboard_AWS_Ver_20_8_25\frontend\build

:: 6. Check logs for errors
type C:\Users\slg\Sea_Level_Dashboard_AWS_Ver_20_8_25\backend\logs\server.log
```

Quick Start Commands

Save this as `quick_start.bat`:

```batch
@echo off
echo Starting Sea Level Dashboard...

cd /d C:\Users\slg\Sea_Level_Dashboard_AWS_Ver_20_8_25\backend
call venv\Scripts\activate
start python local_server.py

timeout /t 5
start http://127.0.0.1:30886

echo Server started. Opening browser...
```

Success Indicators

- ✅ Server starts without errors
- ✅ Can access http://127.0.0.1:30886
- ✅ Can access http://5.102.231.16:30886 from external device
- ✅ API docs load at http://5.102.231.16:30886/docs
- ✅ No firewall blocking messages
- ✅ Frontend loads (if built)

Support Commands

If something goes wrong, run this diagnostic script:

Create `diagnose.bat`:

```batch
@echo off
echo === Sea Level Dashboard Diagnostic ===
echo.
echo Python Version:
python --version
echo.
echo Node Version:
node --version
echo.
echo NPM Version:
npm --version
echo.
echo Port 30886 Status:
netstat -an | findstr :30886
echo.
echo Firewall Rules:
netsh advfirewall firewall show rule name="Sea Level Dashboard"
echo.
echo PostgreSQL Status:
sc query postgresql-x64-13
echo.
echo Current Directory:
cd
echo.
echo Python Packages:
pip list | findstr "fastapi uvicorn sqlalchemy"
echo.
```