How to Run LibreOffice in Docker for Document Conversion

Convert Word, Excel, PowerPoint, and other office documents to PDF and other formats using LibreOffice in Docker containers.

By @nawazdhandala • Feb 08, 2026 • 5 min read

Docker LibreOffice Document Conversion PDF Office Documents Automation

Converting office documents to PDF is a common requirement for web applications. Users upload Word files, spreadsheets, and presentations, and the system needs to render them as PDFs for viewing or archiving. LibreOffice handles this conversion with high fidelity, supporting virtually every office document format. Running LibreOffice in Docker keeps your server clean and gives you a predictable, reproducible conversion environment.

Why LibreOffice in Docker?

LibreOffice is a large application with many dependencies. Installing it on a production server adds bloat and potential security surface area. Docker isolates LibreOffice completely, letting you run it as a stateless conversion service that starts, converts, and exits. You get the full power of LibreOffice's rendering engine without permanently installing it.

Quick Start

After building the custom image below as `lo-converter`, convert a Word document to PDF with a single Docker command:

```bash
# Convert a DOCX file to PDF using LibreOffice in headless mode

docker run --rm \
  -v "$(pwd)/docs:/docs" \
  lo-converter \
  --headless --convert-to pdf --outdir /docs /docs/report.docx
```

The `--headless` flag runs LibreOffice without a graphical interface. The `--convert-to pdf` flag specifies the output format. The file is converted and the PDF is written to the same directory.

Building a Custom Image

For production use, build an image with specific fonts and configuration:

```dockerfile
# Dockerfile - LibreOffice document conversion service
FROM ubuntu:22.04

ENV DEBIAN_FRONTEND=noninteractive

# Install LibreOffice and a comprehensive set of fonts
RUN apt-get update && apt-get install -y --no-install-recommends \
  libreoffice-core \
  libreoffice-writer \
  libreoffice-calc \
  libreoffice-impress \
  fonts-liberation \
  fonts-dejavu \
  fonts-noto \
  fonts-noto-cjk \
  fonts-noto-color-emoji \
  fonts-crosextra-carlito \
  fonts-crosextra-caladea \
  && apt-get clean \
  && rm -rf /var/lib/apt/lists/*

# Create a non-root user for running LibreOffice
RUN useradd -m -s /bin/bash converter
USER converter

WORKDIR /docs

ENTRYPOINT ["libreoffice", "--headless", "--norestore"]
```

Build the image:

```bash
docker build -t lo-converter .
```

The font packages are critical. Without `fonts-crosextra-carlito` and `fonts-crosextra-caladea`, LibreOffice substitutes Calibri and Cambria (common Microsoft fonts) with Liberation fonts, which changes the document layout. Carlito and Caladea are metrically compatible replacements that preserve formatting.

Supported Conversions

LibreOffice supports an extensive list of format conversions. Here are the most common ones:

```bash
# Word document to PDF
docker run --rm -v $(pwd)/docs:/docs lo-converter \
  --convert-to pdf /docs/report.docx

# Excel spreadsheet to PDF
docker run --rm -v $(pwd)/docs:/docs lo-converter \
  --convert-to pdf /docs/data.xlsx

# PowerPoint presentation to PDF
docker run --rm -v $(pwd)/docs:/docs lo-converter \
  --convert-to pdf /docs/slides.pptx

# Word document to HTML
docker run --rm -v $(pwd)/docs:/docs lo-converter \
  --convert-to html /docs/article.docx

# CSV to XLSX (Excel format)
docker run --rm -v $(pwd)/docs:/docs lo-converter \
  --convert-to xlsx /docs/data.csv

# ODT (OpenDocument) to DOCX
docker run --rm -v $(pwd)/docs:/docs lo-converter \
  --convert-to docx /docs/document.odt
```

Building an HTTP Conversion Service

For a web application, wrap LibreOffice in an HTTP API:

```python
# server.py - Flask-based document conversion API
import os
import subprocess
import tempfile
from pathlib import Path
from flask import Flask, request, send_file, jsonify
from werkzeug.utils import secure_filename

app = Flask(__name__)

ALLOWED_EXTENSIONS = {
    'docx', 'doc', 'xlsx', 'xls', 'pptx', 'ppt',
    'odt', 'ods', 'odp', 'rtf', 'csv', 'html'
}

OUTPUT_FORMATS = {'pdf', 'docx', 'xlsx', 'html', 'txt', 'png'}


def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS


@app.route('/convert', methods=['POST'])
def convert():
    """Convert an uploaded document to the specified format."""
    if 'file' not in request.files:
        return jsonify({'error': 'No file uploaded'}), 400

    file = request.files['file']
    target_format = request.form.get('format', 'pdf').lower()

    if not allowed_file(file.filename):
        return jsonify({'error': f'Unsupported file type: {file.filename}'}), 400

    if target_format not in OUTPUT_FORMATS:
        return jsonify({'error': f'Unsupported output format: {target_format}'}), 400

    filename = secure_filename(file.filename)
    if not filename:
        return jsonify({'error': 'Invalid file name'}), 400

    # Save the uploaded file to a temporary directory
    with tempfile.TemporaryDirectory() as tmpdir:
        input_path = os.path.join(tmpdir, filename)
        file.save(input_path)
        profile_dir = os.path.join(tmpdir, 'lo-profile')

        # Run LibreOffice conversion
        result = subprocess.run([
            'libreoffice',
            '--headless',
            '--norestore',
            f'-env:UserInstallation={Path(profile_dir).as_uri()}',
            '--convert-to', target_format,
            '--outdir', tmpdir,
            input_path
        ], capture_output=True, text=True, timeout=120)

        if result.returncode != 0:
            return jsonify({'error': 'Conversion failed', 'details': result.stderr}), 500

        # Find the converted output file
        base_name = os.path.splitext(filename)[0]
        output_path = os.path.join(tmpdir, f'{base_name}.{target_format}')

        if not os.path.exists(output_path):
            return jsonify({'error': 'Output file not found'}), 500

        return send_file(
            output_path,
            as_attachment=True,
            download_name=f'{base_name}.{target_format}'
        )


@app.route('/health')
def health():
    return jsonify({'status': 'ok'})


if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

The Dockerfile for the service:

```dockerfile
# Dockerfile - LibreOffice conversion HTTP service
FROM ubuntu:22.04

ENV DEBIAN_FRONTEND=noninteractive

RUN apt-get update && apt-get install -y --no-install-recommends \
  python3 \
  python3-pip \
  libreoffice-core \
  libreoffice-writer \
  libreoffice-calc \
  libreoffice-impress \
  fonts-liberation \
  fonts-noto \
  fonts-crosextra-carlito \
  fonts-crosextra-caladea \
  && apt-get clean \
  && rm -rf /var/lib/apt/lists/*

WORKDIR /app

COPY requirements.txt .
RUN pip3 install --no-cache-dir -r requirements.txt

COPY server.py .

EXPOSE 5000

CMD ["python3", "server.py"]
```

The Python requirements:

```txt
# requirements.txt
flask==3.1.3
gunicorn==26.0.0
```

Docker Compose Configuration

```yaml
# docker-compose.yml - Document conversion service
services:
  converter:
    build: .
    container_name: doc-converter
    ports:
      - "5000:5000"
    volumes:
      - ./output:/app/output
    environment:
      FLASK_ENV: production
    # Use gunicorn for production with multiple workers
    command: gunicorn --bind 0.0.0.0:5000 --workers 4 --timeout 120 server:app
    restart: unless-stopped
```

Using the Conversion API

Upload and convert documents:

```bash
# Convert a Word document to PDF via the HTTP API
curl -X POST http://localhost:5000/convert \
  -F "file=@report.docx" \
  -F "format=pdf" \
  --output report.pdf
```

Convert a spreadsheet to PDF:

```bash
# Convert an Excel file to PDF
curl -X POST http://localhost:5000/convert \
  -F "file=@financials.xlsx" \
  -F "format=pdf" \
  --output financials.pdf
```

Batch Conversion Script

Convert all files in a directory:

```bash
#!/bin/bash
# scripts/batch-convert.sh - Convert all office documents in a directory to PDF

INPUT_DIR="./docs/input"
OUTPUT_DIR="./docs/output"

mkdir -p "$OUTPUT_DIR"

# Process each supported file format
for file in "$INPUT_DIR"/*.{docx,doc,xlsx,xls,pptx,ppt,odt,ods}; do
  [ -f "$file" ] || continue
  filename=$(basename "$file")

  echo "Converting: $filename"
  docker run --rm \
    -v "$(pwd)/docs:/docs" \
    lo-converter \
    --convert-to pdf \
    --outdir /docs/output \
    "/docs/input/$filename"
done

echo "Batch conversion complete."
```

Handling Concurrent Conversions

LibreOffice is single-threaded per instance. For concurrent conversions, run multiple containers behind a load balancer or use a task queue:

```yaml
# docker-compose.yml - Scaled conversion service
services:
  converter:
    build: .
    command: gunicorn --bind 0.0.0.0:5000 --workers 4 --timeout 120 server:app
    deploy:
      replicas: 3
    networks:
      - convnet

  nginx:
    image: nginx:alpine
    ports:
      - "5000:5000"
    volumes:
      - ./nginx.conf:/etc/nginx/nginx.conf:ro
    depends_on:
      - converter
    networks:
      - convnet

networks:
  convnet:
    driver: bridge
```

Wrapping Up

LibreOffice in Docker provides a reliable, isolated document conversion service that handles Word, Excel, PowerPoint, and dozens of other formats. The key to good output quality lies in installing the right font packages to match the fonts used in the source documents. Whether you use it for one-off conversions via the command line or as a production HTTP service, Docker keeps the setup clean and the results predictable.

Share this article

Nawaz Dhandala Author
@nawazdhandala • Feb 08, 2026 • 5 min read
Nawaz is building OneUptime with a passion for engineering reliable systems and improving observability.
GitHub