Production Deployment Guide

OpenWebUI + ComfyUI Multi-Workflow System

For image.ldmathes.cc (RTX 5090 Beast)

Your Setup Recap

Infrastructure:

- **Machine 1**: OpenWebUI (Windows 11)
- **Machine 2**: Ollama Server (Windows 11)
- **Machine 3**: ComfyUI Server - image.ldmathes.cc (Windows 11, RTX 5090)
- **Network**: All on same LAN
- **Security**: Cloudflare authentication already configured
- **Use Case**: Real-time experimental image generation

Models Available:

- Flux & Flux 2 (dev/schnell)
- Z-Image
- SD 1.5
- SDXL
- SD3

Goal: Multiple specialized workflows accessible through OpenWebUI with automatic routing

---

Architecture Decision

Given your setup (LAN, Cloudflare auth, RTX 5090, real-time gen), we're using:

Workflow Router Middleware Approach:

```
User in OpenWebUI → Cloudflare → Router API (port 8189) → Selects Workflow → ComfyUI (port 8188)
```

Why this works better than direct connection: ✓ Multiple workflows without re-uploading ✓ Automatic task detection (txt2img vs img2img) ✓ Model-specific routing (Flux vs Z-Image vs SDXL) ✓ Parameter validation and optimization ✓ Centralized logging ✓ Easy to extend

---

Phase 1: Setup ComfyUI Machine (image.ldmathes.cc)

Step 1.1: Organize Directory Structure

```powershell
# Run on ComfyUI machine
cd C:\ComfyUI  # or wherever ComfyUI is installed

# Create workflow directory
mkdir workflows
mkdir logs

# Verify models are properly organized
ls models\diffusion_models  # Should show z_image_bf16.safetensors
ls models\text_encoders     # Should show qwen_3_4b.safetensors
ls models\vae               # Should show ae.safetensors
```

IMPORTANT: Based on your Vantage workflow, you need these exact model files:

- `models\diffusion_models\z_image_bf16.safetensors`
- `models\text_encoders\qwen_3_4b.safetensors`
- `models\vae\ae.safetensors`
- Optional: `models\unet\z_image-Q5_K_S.gguf` (for GGUF version)

Step 1.2: Install Python Dependencies

```powershell
# Use ComfyUI's embedded Python (recommended)
cd C:\ComfyUI

# Install FastAPI and dependencies
.\python_embeded\python.exe -m pip install fastapi uvicorn httpx python-multipart

# Verify installation
.\python_embeded\python.exe -c "import fastapi; print('✓ FastAPI installed')"
```

Step 1.3: Deploy Router Files

Copy these files to `C:\ComfyUI\`:

1. `workflow_router.py` - Core routing logic
2. `api_wrapper.py` - FastAPI middleware
3. `convert_to_api.py` - Workflow converter utility

```powershell
# After copying files
dir C:\ComfyUI\*.py
# Should show: workflow_router.py, api_wrapper.py, convert_to_api.py
```

Step 1.4: Convert Your Vantage Workflow

```powershell
cd C:\ComfyUI

# Convert to API format
.\python_embeded\python.exe convert_to_api.py `
  "C:\path\to\Vantage-Z-Image.json" `
  "workflows\vantage-z-image_api.json"

# Verify output
dir workflows\
# Should show: vantage-z-image_api.json
```

Critical Node IDs (from your workflow):

- **Positive Prompt**: Node 42 (CLIPTextEncode)
- **Negative Prompt**: Node 56 (CLIPTextEncode)
- **Sampler**: Node 41 (KSampler)
- **Save Image**: Node 9 (SaveImage)

Step 1.5: Create Startup Scripts

File: `C:\ComfyUI\start-comfyui.bat`

```batch
@echo off
echo ========================================
echo Starting ComfyUI Server
echo ========================================
cd /d C:\ComfyUI

REM Start ComfyUI with optimal settings for RTX 5090
python_embeded\python.exe -s ComfyUI\main.py ^
  --listen 0.0.0.0 ^
  --port 8188 ^
  --preview-method auto ^
  --highvram ^
  --disable-auto-launch

pause
```

File: `C:\ComfyUI\start-router.bat`

```batch
@echo off
echo ========================================
echo Starting Workflow Router API
echo ========================================
cd /d C:\ComfyUI

REM Start router on port 8189
python_embeded\python.exe api_wrapper.py ^
  --host 0.0.0.0 ^
  --port 8189 ^
  --comfyui-url http://localhost:8188 ^
  --workflows-dir workflows

pause
```

File: **`C:\ComfyUI\start-all.bat`** (Convenience script)

```batch
@echo off
echo ========================================
echo Starting Complete System
echo ========================================

REM Start ComfyUI in new window
start "ComfyUI Server" cmd /k "cd /d C:\ComfyUI && start-comfyui.bat"

REM Wait for ComfyUI to start
echo Waiting for ComfyUI to initialize...
timeout /t 10

REM Start Router in new window
start "Workflow Router" cmd /k "cd /d C:\ComfyUI && start-router.bat"

echo.
echo ========================================
echo All services started!
echo ========================================
echo.
echo ComfyUI:     http://localhost:8188
echo Router API:  http://localhost:8189
echo.
echo Configure OpenWebUI with:
echo   COMFYUI_BASE_URL=http://image.ldmathes.cc:8189/
echo.
pause
```

Step 1.6: Test Router Independently

```powershell
# In ComfyUI directory
.\python_embeded\python.exe workflow_router.py --list

# Should output:
# === Available Workflows ===
# • vantage-z-image
#   Type: txt2img
#   Model: z-image
#   Requires Image: False
#   Nodes: positive_prompt, negative_prompt, sampler, save_image
```

Phase 2: Configure OpenWebUI Machine

Step 2.1: Update Docker Compose (If using Docker)

File: `docker-compose.yml` on OpenWebUI machine

```yaml
version: '3.8'

services:
  openwebui:
    image: ghcr.io/open-webui/open-webui:main
    container_name: openwebui
    ports:
      - "3000:8080"
    environment:
      # Ollama connection (replace with actual IP)
      - OLLAMA_BASE_URL=http://192.168.1.XXX:11434

      # ComfyUI via Router (CRITICAL: Use port 8189, not 8188!)
      - COMFYUI_BASE_URL=http://image.ldmathes.cc:8189/

      # Enable image generation
      - ENABLE_IMAGE_GENERATION=True

      # Optional: Set default parameters
      - DEFAULT_IMAGE_WIDTH=1024
      - DEFAULT_IMAGE_HEIGHT=1024

    volumes:
      - openwebui_data:/app/backend/data

    restart: unless-stopped

    # Allow Docker to resolve LAN hostnames
    extra_hosts:
      - "image.ldmathes.cc:192.168.1.XXX"  # Replace with actual IP

volumes:
  openwebui_data:
```

Replace placeholders:

- `192.168.1.XXX` with actual machine IPs
- Verify Cloudflare isn't blocking port 8189

Step 2.2: Restart OpenWebUI

```powershell
# On OpenWebUI machine
docker-compose down
docker-compose up -d

# Check logs
docker logs -f openwebui
```

Phase 3: Configuration & Testing

Step 3.1: Start Services in Order

On ComfyUI machine (image.ldmathes.cc):

```powershell
# Option A: Start all at once
cd C:\ComfyUI
.\start-all.bat

# Option B: Start individually (for debugging)
# Terminal 1:
.\start-comfyui.bat

# Terminal 2 (wait 10 seconds):
.\start-router.bat
```

Verify services are running:

```powershell
# Test ComfyUI
curl http://localhost:8188/system_stats

# Test Router
curl http://localhost:8189/
# Should return: {"status": "running", "workflows": 1, ...}

# Test workflow listing
curl http://localhost:8189/workflows
```

Step 3.2: Configure OpenWebUI UI

1. **Access OpenWebUI**: `http://[openwebui-machine-ip]:3000`
2. **Navigate to Settings**:
   - Click profile/avatar (top-right)
   - Admin Panel → Settings → Images
3. **Configure Image Generation**:
   - **Image Generation Engine**: ComfyUI
   - **ComfyUI API URL**: `http://image.ldmathes.cc:8189/`
     - ⚠️**Port 8189** (router), NOT 8188 (direct ComfyUI)
   - **Enable Image Generation**: ON
4. **Test Connection**:
   - Should show green checkmark
   - If fails, check:
     - Router is running on 8189
     - Firewall allows 8189
     - URL is correct (with trailing slash)

Step 3.3: First Generation Test

In OpenWebUI chat:

```
Generate a realistic photograph of a sunset over mountains
```

What happens behind the scenes:

1. OpenWebUI sends request to router (8189)
2. Router analyzes: "txt2img task detected"
3. Router selects: vantage-z-image workflow
4. Router injects: your prompt into node 42
5. Router forwards to ComfyUI (8188)
6. ComfyUI generates using Z-Image model
7. Image returns to OpenWebUI

Expected result: Image appears in chat (~2-5 seconds on RTX 5090)

Check logs:

```powershell
# On ComfyUI machine
type logs\api_wrapper.log

# Should show:
# [timestamp] - INFO - Received prompt request: Generate a realistic...
# [timestamp] - INFO - Selected workflow: vantage-z-image (task: txt2img)
# [timestamp] - INFO - Successfully queued prompt: [prompt_id]
```

Phase 4: Add More Workflows

Step 4.1: Workflow Creation Process

For each new workflow (img2img, inpaint, etc.):

1. **Design in ComfyUI**:
   - Open ComfyUI UI: `http://image.ldmathes.cc:8188`
   - Build workflow
   - Test it works
   - **Critical**: Use consistent node IDs for prompts/samplers
2. **Standardize Node IDs** (IMPORTANT):

```
Always use these node IDs across ALL workflows:
- Node 42: Positive prompt (CLIPTextEncode)
- Node 56: Negative prompt (CLIPTextEncode)
- Node 41: Sampler (KSampler)
- Node 9: Save Image (SaveImage)
```

This ensures router can inject parameters consistently.

3. **Export API Format**:
   - Settings → Enable Dev Mode
   - Save (API Format) → `workflow_name.json`
4. **Convert and Deploy**:

```powershell
cd C:\ComfyUI

.\python_embeded\python.exe convert_to_api.py `
  "workflow_name.json" `
  "workflows\workflow_name_api.json"
```

5. **Restart Router**:

```powershell
# Router auto-loads new workflows, but restart ensures clean state
# In router terminal: Ctrl+C
.\start-router.bat
```

6. **Test**:

```powershell
curl http://localhost:8189/workflows
# Should list new workflow
```

Step 4.2: Example: Creating img2img Workflow

Design requirements for img2img:

```
Required nodes:
- LoadImage node (for user upload)
- VAEEncode (convert image to latent)
- CLIPTextEncode x2 (positive/negative)
- KSampler (with denoise < 1.0)
- VAEDecode
- SaveImage

Node IDs must be:
- 42: Positive prompt
- 56: Negative prompt
- 41: Sampler
- 9: Save
```

Router will auto-detect img2img when:

- User attaches an image in OpenWebUI
- Prompt contains keywords like "make this", "transform", "style"

Step 4.3: Workflow Naming Convention

```
Naming format: [task]-[model]-[variant]_api.json

Examples:
- txt2img-z-image-quality_api.json
- txt2img-flux-dev_api.json
- img2img-sdxl-standard_api.json
- inpaint-sd15-advanced_api.json
- upscale-realesrgan_api.json
```

Router will:

- Parse task type from filename
- Auto-detect model from workflow
- Select best match for user request

Phase 5: Optimization for RTX 5090

Step 5.1: Optimal Z-Image Parameters

Your Vantage workflow uses:

- **Steps**: 28 (good for quality)
- **CFG**: 4 (good for Z-Image)
- **Sampler**: res_multistep (optimal)
- **Scheduler**: simple

For faster experimentation:

Create variant: `vantage-z-image-fast_api.json`

- Steps: 10-15 (still good quality)
- Everything else same

For maximum quality:

Create variant: `vantage-z-image-quality_api.json`

- Steps: 40-50
- Add upscaling nodes

Step 5.2: Model-Specific Settings

Flux 2 Dev/Schnell:

```
Steps: 20-28 (dev), 4-8 (schnell)
CFG: 3.5-5.0
Sampler: euler
```

SDXL:

```
Steps: 25-35
CFG: 7-9
Sampler: dpmpp_2m or euler_a
```

SD 1.5:

```
Steps: 20-30
CFG: 7-11
Sampler: dpmpp_2m_sde
```

Step 5.3: Memory Optimization

With RTX 5090's 32GB VRAM, you can:

1. **Load multiple models** (model switching in workflow)
2. **Enable highvram mode** (already in start-comfyui.bat)
3. **Run batches** if needed
4. **Use higher resolutions** (2048x2048+)

Update start-comfyui.bat for maximum performance:

```batch
python_embeded\python.exe -s ComfyUI\main.py ^
  --listen 0.0.0.0 ^
  --port 8188 ^
  --preview-method auto ^
  --highvram ^
  --reserve-vram 2 ^
  --disable-auto-launch ^
  --cuda-malloc
```

Phase 6: Monitoring & Maintenance

Step 6.1: Enable Logging

Already configured in api_wrapper.py, logs go to:

- `C:\ComfyUI\logs\api_wrapper.log`

Monitor logs:

```powershell
# Real-time monitoring
Get-Content C:\ComfyUI\logs\api_wrapper.log -Wait -Tail 50

# Search for errors
Select-String "ERROR" C:\ComfyUI\logs\api_wrapper.log

# Check workflow routing
Select-String "Selected workflow" C:\ComfyUI\logs\api_wrapper.log
```

Step 6.2: Performance Monitoring

Create: `C:\ComfyUI\monitor.ps1`

```powershell
# Simple monitoring script
while ($true) {
    Clear-Host
    Write-Host "===== ComfyUI System Monitor =====" -ForegroundColor Cyan
    Write-Host ""

    # Check if services are running
    $comfyui = Test-NetConnection -ComputerName localhost -Port 8188 -WarningAction SilentlyContinue
    $router = Test-NetConnection -ComputerName localhost -Port 8189 -WarningAction SilentlyContinue

    Write-Host "ComfyUI (8188): " -NoNewline
    if ($comfyui.TcpTestSucceeded) {
        Write-Host "RUNNING" -ForegroundColor Green
    } else {
        Write-Host "DOWN" -ForegroundColor Red
    }

    Write-Host "Router (8189):  " -NoNewline
    if ($router.TcpTestSucceeded) {
        Write-Host "RUNNING" -ForegroundColor Green
    } else {
        Write-Host "DOWN" -ForegroundColor Red
    }

    Write-Host ""

    # GPU stats (requires nvidia-smi)
    Write-Host "GPU Status:" -ForegroundColor Cyan
    nvidia-smi --query-gpu=utilization.gpu,memory.used,memory.total,temperature.gpu --format=csv,noheader,nounits

    Write-Host ""
    Write-Host "Press Ctrl+C to exit" -ForegroundColor Gray

    Start-Sleep -Seconds 5
}
```

Run monitor:

```powershell
powershell -ExecutionPolicy Bypass -File C:\ComfyUI\monitor.ps1
```

Step 6.3: Automatic Restart (Optional)

Create Windows Task Scheduler task:

1. Open Task Scheduler
2. Create Basic Task: "ComfyUI Auto-Start"
3. Trigger: At system startup
4. Action: Start program
    - Program: `C:\ComfyUI\start-all.bat`
    - Start in: `C:\ComfyUI`

Troubleshooting Guide

Issue: Router can't find workflows

Symptoms:

```
ERROR: No workflows available!
```

Fix:

```powershell
cd C:\ComfyUI
dir workflows\*_api.json

# If empty, convert your workflows:
.\python_embeded\python.exe convert_to_api.py `
  "Vantage-Z-Image.json" `
  "workflows\vantage-z-image_api.json"
```

Issue: OpenWebUI can't connect

Symptoms: "Connection failed" in OpenWebUI settings

Check:

```powershell
# 1. Is router running?
curl http://localhost:8189/

# 2. Is port accessible from OpenWebUI machine?
# On OpenWebUI machine:
curl http://image.ldmathes.cc:8189/

# 3. Check Windows Firewall
netsh advfirewall firewall show rule name="ComfyUI Router"

# If rule doesn't exist, create it:
netsh advfirewall firewall add rule name="ComfyUI Router" ^
  dir=in action=allow protocol=TCP localport=8189
```

Issue: Generations fail silently

Symptoms: Request queued but no image generated

Debug:

```powershell
# Check ComfyUI console for errors
# Look for "Model not found" or "Node error"

# Check router logs
type logs\api_wrapper.log | Select-String "ERROR"

# Test workflow manually in ComfyUI UI
# Load workflow_api.json and try to queue
```

Issue: Wrong workflow selected

Symptoms: Z-Image used when you wanted Flux

Fix: Add preference parameter (need to implement in OpenWebUI request)

Temporary workaround:

```python
# Edit workflow_router.py, in select_workflow():
# Add debug logging:
logger.info(f"Task: {task_type}, Candidates: {candidates}")

# This shows scoring for each workflow
```

Issue: Slow generation on RTX 5090

Should NOT happen, but if it does:

```powershell
# Check GPU usage
nvidia-smi dmon -s u

# Should show ~90-100% GPU util during gen

# If low GPU usage:
# 1. Check CUDA installation
# 2. Verify ComfyUI using GPU:
#    Look in ComfyUI console for "cuda" not "cpu"
# 3. Try --cuda-malloc flag in start script
```

Next Steps: Complex Workflow Integration

You mentioned wanting to "web-ize" a complex workflow next. Here's the process:

For Your Next Complex Workflow:

1. **Share the workflow file** (like you did with Vantage)
2. **I'll analyze**:
    - Node structure
    - Input/output requirements
    - Parameters that need injection
    - Model dependencies
3. **We'll discuss**:
    - Can it be automated?
    - What parameters should OpenWebUI control?
    - Should it be one workflow or split into variants?
4. **I'll provide**:
    - Converted API format
    - Custom routing logic if needed
    - Parameter mapping
    - Testing instructions

Questions to Answer for Complex Workflows:

- Does it need image input? (img2img, inpaint)
- Multiple images? (img2img batch)
- Mask required? (inpainting)
- What parameters should users control?
- Should different parts be separate workflows?
- Model switching needed?
- ControlNet integration?

Summary of What You Now Have

✅**Multi-workflow system** with automatic routing ✅**Z-Image workflow** ready for txt2img ✅**Extensible architecture** for adding more workflows ✅**Proper logging** and monitoring ✅**LAN-optimized** for your setup ✅**RTX 5090 optimized** for speed ✅**Cloudflare compatible** with your existing auth

What's Ready:

- Convert ANY ComfyUI workflow to API format
- Add it to workflows/ directory
- Router automatically picks it up
- OpenWebUI uses it seamlessly

Current Capabilities:

- Txt2img with Z-Image model
- Parameter control (prompt, steps, cfg, size)
- Real-time generation
- Progress tracking

Ready to Add:

- Img2img workflows
- Inpainting
- Upscaling
- Flux 2 variants
- SDXL workflows
- Your complex experimental workflows

Quick Reference Commands

Start Everything:

```powershell
cd C:\ComfyUI
.\start-all.bat
```

Check Status:

```powershell
curl http://localhost:8188/system_stats  # ComfyUI
curl http://localhost:8189/              # Router
curl http://localhost:8189/workflows     # List workflows
```

Add New Workflow:

```powershell
.\python_embeded\python.exe convert_to_api.py `
  "new-workflow.json" `
  "workflows\new-workflow_api.json"
```

Monitor:

```powershell
Get-Content logs\api_wrapper.log -Wait -Tail 50
```

You're now ready to generate! 🎨

Send me your complex workflow when you're ready for step 2, and I'll analyze how to web-ize it properly.