📘 MASTER DOCUMENT Single-PC Open-Source AI Stack – Everything in One Place

Version 2025-08-24-A | 24 Aug 2025

📑 Preface – Now with Multi-Voice TTS and VS Code Integration

This revised document unifies **11 flagship open-source AI projects** on a single PC, with **MeloTTS** replacing Orpheus-TTS for high-quality, multilingual, multi-speaker, zero-shot voice cloning. It includes Visual Studio Code (VS Code) setup instructions for seamless development across all projects. You will find:

1. A concise **purpose statement** for every project.
2. Exact **hardware & software requirements** for desktop use.
3. **Step-by-step install & wiring instructions** so every tool talks to every other tool.
4. A **zero-touch weekly update script** that keeps models, containers, and Python packages current.
5. A **Docker-vs-native decision matrix** with GPU passthrough notes.
6. **VS Code configuration** for managing, debugging, and running the entire stack.

Nothing has been omitted; everything is copy-paste ready.

📋 Table of Contents

🔍 Section A – Crystal-Clear Reference Table *(11 projects, nothing omitted)*

#	Project	Essence (single-sentence North-Star)	Homepage	Run on PC (typical)	Integration re
1	DeepSeek-R1	Fully open reasoning LLM rivaling OpenAI-o1 on math/code via MIT weights & OpenAI-compatible API.	https://deepseek.com	7–8 B 8-bit ≈ 8–10 GB VRAM; 70 B ≥ 48 GB VRAM or CPU-offload.	`vllm serve DeepSeek-R1` or `c` tools to `http://localhost:800`
2	Ollama	"Docker-for-LLMs" — single-command pull & run of any open model locally.	https://ollama.ai	7–13 B models run in 8–16 GB RAM; CPU fallback.	Expose `http://localhost:114` tool treats it like GPT-4.
3	OpenManus	Visual no-code workbench chaining LLMs, tools & data into autonomous AI apps.	https://github.com/FoundationAgents/OpenManus	Docker-Compose; 4–8 GB RAM; GPU optional.	UI lets you pick `localhost:1143` service.
4	LangChain	Universal "LEGO kit" for building LLM apps via composable prompts, memory, retrieval & agents.	https://langchain.com	Pure Python/JS; 2–4 GB RAM baseline.	Import `ChatOllama(base_url="` with any endpoint.
5	AutoGen (Microsoft)	Lightweight multi-agent framework that spins up LLM "teams" to negotiate tasks.	https://microsoft.github.io/autogen	`pip install pyautogen`; any OpenAI-compatible endpoint.	Set `llm_config={"base_url":"http://localho`
6	OpenSora	Open, distributed training platform turning any cluster into a generative-	https://github.com/hpcaitech/Open-Sora	Linux + CUDA; dev mode ≈ 12 GB VRAM.	Export HF checkpoint → `ollama`

#	Project	Essence (single-sentence North-Star)	Homepage	Run on PC (typical)	Integration re
		model factory.			
7	Haystack (deepset)	End-to-end semantic search & RAG framework for production-grade document workflows.	https://haystack.deepset.ai	CPU baseline; GPU optional.	Use `OpenAIGenerator(api_bas` share vector DB across LangChain
8	Text-Generation-WebUI	Browser dashboard for downloading, chatting & serving open LLMs locally or via API.	https://github.com/oobabooga/text-generation-webui	4 GB VRAM for 7 B, 12 GB for 30 B.	Enable "OpenAI-compatible API"
9	Whisper (OpenAI)	State-of-the-art offline speech-to-text & translation in 99 languages.	https://github.com/openai/whisper	CPU real-time; GPU 10× faster.	Wrap with FastAPI → `http://lo` any pipeline.
10	Letta *(ex-MemGPT)*	Persistent, portable memory layer for LLM agents across sessions & frameworks.	https://letta.ai	`pip install letta` ; 2–4 GB RAM; any local LLM endpoint.	`letta server --model-endpo` agents stored at `http://localh`
11	**MeloTTS** *(new)*	Apache-2.0 multilingual, multi-speaker, zero-shot voice-cloning TTS (~500M–2B models).	https://github.com/myshell-ai/MeloTTS	500M CPU-only; 1–2B ≈ 4–8 GB VRAM.	`uvicorn melo_tts_api:app --` `http://localhost:8001/v1/au`

Note: MeloTTS replaces Orpheus-TTS, offering similar functionality with lower resource requirements and broader language support.

RAG Explanation:

RAG (Retrieval-Augmented Generation) enhances LLMs by retrieving external information for more accurate, grounded responses.

- **How it works**: Combines information retrieval (e.g., search engines) with LLMs to fetch relevant data and generate informed answers.
- **Benefits**:
  - **Accuracy**: Reduces hallucinations with factual, up-to-date data.
  - **Relevance**: Delivers contextually specific answers.
  - **Up-to-Date Information**: Connects to live data sources.
  - **Source Attribution**: Provides citations for trust.

🛠️ Section B – Complete User Manual

All commands assume you are inside `~/AI_STACK` (Windows: `C:\Users\<YourUser>\AI_STACK`).

B-0 Prep & Safety

```bash
mkdir ~/AI_STACK && cd ~/AI_STACK
# Install: Git, Python 3.10+, Docker Desktop, NVIDIA Container Toolkit, CUDA 12.x
# Windows: Use PowerShell as Administrator; see Appendix for setup-ai-stack.ps1
```

B-1 DeepSeek-R1

```bash
ollama run deepseek-r1:7b
```

B-2 Ollama

```bash
ollama pull llama3.1:8b
ollama serve # http://localhost:11434/v1
```

B-3 OpenManus

```bash
git clone https://github.com/FoundationAgents/OpenManus.git
cd OpenManus
docker-compose up
# http://localhost:3000
# In UI set OLLAMA_URL to http://host.docker.internal:11434
```

B-4 LangChain

```bash
python -m venv lc-env && source lc-env/bin/activate
pip install langchain langchain-community
```

B-5 AutoGen

```bash
pip install pyautogen
```

B-6 OpenSora

```bash
docker pull hpcaitech/opensora:latest
docker run --gpus all -it -v $PWD/data:/workspace/data hpcaitech/opensora bash
```

B-7 Haystack

```bash
pip install haystack-ai sentence-transformers faiss-cpu
```

B-8 Text-Generation-WebUI

```bash
git clone https://github.com/oobabooga/text-generation-webui.git tg-webui
cd tg-webui
./start_linux.sh   # or start_windows.bat
# http://localhost:5000 → enable OpenAI API
```

B-9 Whisper

```bash
pip install openai-whisper fastapi uvicorn
```

Create `whisper_api.py` (see Section A for integration) and:

```bash
uvicorn whisper_api:app --host 0.0.0.0 --port 9000
```

B-10 Letta

```bash
pip install letta
letta server --model-endpoint http://localhost:11434/v1 --model llama3.1:8b
```

B-11 MeloTTS (new section)

1. **Install**

   ```bash
   pip install melotts
   python -c "from MeloTTS.melo.download_utils import download_models; download_models()"
   # Optional GPU support: Ensure CUDA and torch are installed
   ```

2. **Serve**

   Create `melo_tts_api.py`:

   ```python
   from fastapi import FastAPI
   from pydantic import BaseModel
   from melo.api import TTS
   import torch
   import soundfile as sf
   import io

   app = FastAPI()

   class TTSRequest(BaseModel):
       text: str
       voice: str = "EN-US"  # Default to US English voice

   @app.post("/v1/audio/speech")
   async def generate_speech(request: TTSRequest):
       device = "cuda" if torch.cuda.is_available() else "cpu"
       model = TTS(language="EN", device=device)  # Adjust language as needed
       speaker_ids = model.hps.data.spk2id
       speaker = request.voice if request.voice in speaker_ids else "EN-US"
       output = io.BytesIO()
       audio = model.tts_to_file(request.text, speaker_ids[speaker], None, speed=1.0)
       sf.write(output, audio, model.hps.data.sampling_rate, format="WAV")
       return {"audio": output.getvalue()}
   ```

   Run:

   ```bash
   uvicorn melo_tts_api:app --host 0.0.0.0 --port 8001
   ```

3. **Quick Voice Test**

```bash
curl -X POST http://localhost:8001/v1/audio/speech \
    -H "Content-Type: application/json" \
    -d '{"text":"Hello world!","voice":"EN-US"}' \
    --output hello.wav
```

4. **Integration**
    - Use OpenAI-compatible client pointing to `http://localhost:8001/v1/audio/speech`.
    - Supports voices like `EN-US`, `EN-UK`, `ES`, `FR`, etc. Check `model.hps.data.spk2id` for available speakers.
    - Tag multi-speaker text with identifiers or non-verbals (e.g., `(laughs)`).

B-12 End-to-End Mini Use-Case (now includes MeloTTS)

1. Record `question.wav` (e.g., "What is the capital of France?").
2. Transcribe: `curl -X POST -F file=@question.wav http://localhost:9000/transcribe`
3. Send transcript → LangChain → RAG → DeepSeek-R1 → Letta memory.
4. Feed final answer to MeloTTS → instant spoken response:

```bash
curl -X POST http://localhost:8001/v1/audio/speech -d '{"text":"The capital of France is Paris.","voice":"EN-US"}'
--output answer.wav
```

B-13 Daily Startup Cheat-Sheet

```bash
ollama serve &
uvicorn whisper_api:app --port 9000 &
letta server &
uvicorn melo_tts_api:app --host 0.0.0.0 --port 8001 &
docker-compose -f OpenManus/docker-compose.yml up
# Run in VS Code tasks for easier management (see B-15)
```

B-14 Troubleshooting Quick Fixes

- **Port collision**: Change `--port` or `.env` (e.g., 11434, 5000, 8001, 9000).
- **GPU OOM**: Use 4-bit/8-bit quantized models (e.g., `ollama pull llama3.1:8b-q4_0`).
- **Docker GPU issues**: Reinstall NVIDIA Container Toolkit.
- **Python package errors**: Reinstall dependencies:

```bash
pip install langchain langchain-community pyautogen haystack-ai sentence-transformers faiss-cpu openai-whisper
fastapi uvicorn letta melotts
```

B-15 VS Code Setup for All Projects (new section)

1. **Install VS Code**

```powershell
winget install Microsoft.VisualStudioCode --silent --accept-package-agreements
```

2. **Install Extensions**
    - Python (`ms-python.python`)
    - Jupyter (`ms-toolsai.jupyter`)
    - Docker (`ms-azuretools.vscode-docker`)
    - GitLens (`eamodio.gitlens`)
    - Pylance (`ms-python.vscode-pylance`)
    - REST Client (`humao.rest-client`)

3. **Configure VS Code**
   - Open `C:\Users\<YourUser>\AI_STACK` in VS Code.
   - Select Python interpreter: `Ctrl+Shift+P` → `Python: Select Interpreter` → `C:\Users\<YourUser>\AI_STACK\venv\Scripts\python.exe`.
   - Add to `settings.json` (`Ctrl+,` → Open Settings (JSON)):

```json
{
  "python.defaultInterpreterPath": "C:\\Users\\<YourUser>\\AI_STACK\\venv\\Scripts\\python.exe",
  "terminal.integrated.defaultProfile.windows": "PowerShell",
  "docker.commands.composeUp": "docker-compose -f ${workspaceFolder}\\ai-stack-tts.yml up -d",
  "python.linting.enabled": true,
  "python.linting.pylintEnabled": true
}
```

4. **Create Tasks**

Create `C:\Users\<YourUser>\AI_STACK\.vscode\tasks.json`:

```json
{
  "version": "2.0.0",
  "tasks": [
    {
      "label": "Start Ollama",
      "type": "shell",
      "command": "ollama serve",
      "group": "build",
      "problemMatcher": []
    },
    {
      "label": "Start Text-Gen-WebUI",
      "type": "shell",
      "command": ".\\tg-webui\\start_windows.bat",
      "group": "build",
      "problemMatcher": []
    },
    {
      "label": "Start Whisper API",
      "type": "shell",
      "command": "uvicorn whisper_api:app --host 0.0.0.0 --port 9000",
      "group": "build",
      "problemMatcher": []
    },
    {
      "label": "Start Letta",
      "type": "shell",
      "command": "python -m letta server --model-endpoint http://localhost:11434/v1 --model llama3.1:8b",
      "group": "build",
      "problemMatcher": []
    },
    {
      "label": "Start MeloTTS",
      "type": "shell",
      "command": "uvicorn melo_tts_api:app --host 0.0.0.0 --port 8001",
      "group": "build",
      "problemMatcher": []
    },
    {
      "label": "Start Docker Services",
      "type": "shell",
      "command": "docker-compose -f ai-stack-tts.yml up -d",
      "group": "build",
      "problemMatcher": []
    }
  ]
}
```

5. **Debugging**

   Create `C:\Users\<YourUser>\AI_STACK\.vscode\launch.json`:

```json
{
  "version": "0.2.0",
  "configurations": [
    {
      "name": "Debug Whisper API",
      "type": "python",
      "request": "launch",
      "program": "${workspaceFolder}\\whisper_api.py",
      "args": ["--host", "0.0.0.0", "--port", "9000"],
      "console": "integratedTerminal"
    },
    {
      "name": "Debug MeloTTS API",
      "type": "python",
      "request": "launch",
      "program": "${workspaceFolder}\\melo_tts_api.py",
      "args": ["--host", "0.0.0.0", "--port", "8001"],
      "console": "integratedTerminal"
    }
  ]
}
```

🔄 Section C – Automated Weekly Update Procedure

(covers all 11 projects)

1. **Create script** `update_ai_stack.sh` / `.ps1`

   Windows PowerShell (`update_ai_stack.ps1`):

```powershell
$baseDir = "$env:USERPROFILE\AI_STACK"
$logFile = "$baseDir\logs\update.log"
function Log { param($Msg) Write-Host $Msg -ForegroundColor Cyan; Add-Content $logFile "$(Get-Date) $Msg" }
Log "Updating AI Stack..."
.\venv\Scripts\Activate.ps1
pip install --upgrade langchain langchain-community pyautogen haystack-ai sentence-transformers faiss-cpu openai-whisper fastapi uvicorn letta melotts
python -c "from MeloTTS.melo.download_utils import download_models; download_models()"
ollama pull llama3.1:8b
ollama pull deepseek-r1:7b
git -C "$baseDir\tg-webui" pull
git -C "$baseDir\OpenManus" pull
docker-compose -f "$baseDir\ai-stack-tts.yml" pull
Log "Update complete."
```

2. **Schedule**

   Linux/mac:

```bash
crontab -e
0 3 * * 0 /home/$USER/AI_STACK/update_ai_stack.sh >> ~/AI_STACK/logs/cron.log 2>&1
```

   Windows (Admin PS):

```powershell
schtasks /create /tn "AI_Stack_Weekly_Update" /tr "powershell.exe -File %USERPROFILE%\AI_STACK\update_ai_stack.ps1" /sc weekly /d SUN /st 03:00
```

3. **Roll-back**
    - Models: `ollama list` → `ollama rm <tag>`
    - Python: `pip freeze > requirements.lock`
    - Docker: Tag stable images manually

🐳 Section D – Docker vs. Native Decision Guide

(updated for 11 projects)

Project	Docker?	Native?	When to Docker	When to Stay Native
DeepSeek-R1	✅ via ollama image	✅ native binary	Reproducible cluster	Bare-metal speed
Ollama	❌ single binary	✅ easiest	Dependency isolation	GPU passthrough tricky
OpenManus	✅ compose	❌ complex	Always	Avoid Node/npm hell
LangChain	✅ images	✅ pip	Repro labs	Notebooks
AutoGen	✅ community	✅ pip	CI/CD	Simple scripts
OpenSora	✅ official	❌ Linux only	Non-Linux host	Bare-metal speed
Haystack	✅ official	✅ pip	Production	Dev notebooks
Text-Gen-WebUI	✅ one-liner	✅ scripts	GPU passthrough easy	Custom CUDA
Whisper	✅ small	✅ pip	Clean env	CPU quick test
Letta	✅ community	✅ pip	Multi-service	Single agent
MeloTTS	✅ custom image	✅ pip	GPU passthrough easy	Dev notebooks

GPU Passthrough & Compose

Linux `/etc/docker/daemon.json`:

```json
{
  "default-runtime": "nvidia",
  "runtimes": {
    "nvidia": {
      "path": "nvidia-container-runtime",
      "runtimeArgs": []
    }
  }
}
```

Windows/WSL2 test:

```bash
docker run --rm --gpus all nvidia/cuda:12.2.0-base-ubuntu20.04 nvidia-smi
```

Sample Full-Stack Compose (`ai-stack-tts.yml`)

```yaml
version: "3.9"
services:
  ollama:
    image: ollama/ollama:latest
    ports: ["11434:11434"]
    volumes: ["ollama:/root/.ollama"]
    deploy:
      resources:
        reservations:
          devices:
            - driver: nvidia
              count: 1
              capabilities: [gpu]
  openmanus:
    image: foundationagents/openmanus:latest
    ports: ["3000:3000"]
    environment:
      OLLAMA_URL: "http://host.docker.internal:11434"
  melo-tts:
    build:
      context: .
      dockerfile: Dockerfile
    ports: ["8001:8001"]
    deploy:
      resources:
        reservations:
          devices:
            - driver: nvidia
              count: 1
              capabilities: [gpu]
volumes:
  ollama: {}
```

Create `Dockerfile` for MeloTTS:

```dockerfile
FROM python:3.10-slim
RUN pip install melotts torch torchaudio soundfile fastapi uvicorn
RUN python -c "from MeloTTS.melo.download_utils import download_models; download_models()"
COPY melo_tts_api.py /app/melo_tts_api.py
WORKDIR /app
CMD ["uvicorn", "melo_tts_api:app", "--host", "0.0.0.0", "--port", "8001"]
```

Launch:

```bash
docker-compose -f ai-stack-tts.yml up -d
```

Disk Hygiene

- `docker system prune -f` weekly
- `ollama rm <unused>` after each update
- `pip cache purge` inside venvs

Appendix: PowerShell Automation Script

Below is the updated `setup-ai-stack.ps1`, fixed to resolve the Python package installation error and include MeloTTS.

powershell

```powershell
# ================================================================================
#  setup-ai-stack.ps1  |  Windows 11   |  Elevated PowerShell 7+  (Run as Admin)
#  Installs & starts:  DeepSeek-R1, Ollama, OpenManus, LangChain, AutoGen,
#                      OpenSora (dev), Haystack, Text-Gen-WebUI, Whisper,
#                      Letta, MeloTTS
# ================================================================================

param(
    [switch]$SkipDocker,     # pass -SkipDocker if you want only native installs
    [switch]$Uninstall       # pass -Uninstall to remove everything
)

$ErrorActionPreference = "Stop"
$ProgressPreference = "SilentlyContinue"
$baseDir = "$env:USERPROFILE\AI_STACK"
$logFile = "$baseDir\setup.log"

# ------------------------------------------------------------
#  Helper
# ------------------------------------------------------------
function Log { param($Msg) Write-Host $Msg -ForegroundColor Cyan; Add-Content $logFile "$(Get-Date)  $Msg" }

# ------------------------------------------------------------
#  Uninstall switch
# ------------------------------------------------------------
if ($Uninstall) {
    Log "🧹 Uninstalling AI Stack..."
    try {
        wsl --unregister docker-desktop-data 2>$null
        docker-compose -f "$baseDir\OpenManus\docker-compose.yml" down --remove-orphans 2>$null
        docker-compose -f "$baseDir\ai-stack-tts.yml" down --remove-orphans 2>$null
    } catch {}
    Remove-Item -Recurse -Force $baseDir -ErrorAction SilentlyContinue
    Log "✅ Removed $baseDir"
    exit 0
}

# ------------------------------------------------------------
#  0. Prereqs
# ------------------------------------------------------------
Log "🪟 Ensuring Windows 11 prereqs..."
if (-not ([Security.Principal.WindowsPrincipal]
[Security.Principal.WindowsIdentity]::GetCurrent()).IsInRole([Security.Principal.WindowsBuiltInRole]::Administrator)) {
    Log "❗ Run as Administrator"; exit 1
}

# winget check
if (-not (Get-Command winget -ErrorAction SilentlyContinue)) {
    Log "🧱 Installing App Installer (winget)..."
    Start-Process "ms-windows-store://pdp/?PFN=Microsoft.DesktopAppInstaller_8wekyb3d8bbwe" -Wait
}

# WSL2 + Docker Desktop
if (-not $SkipDocker) {
    if (-not (Get-Command docker -ErrorAction SilentlyContinue)) {
        Log "🐬 Installing Docker Desktop..."
        winget install Docker.DockerDesktop --silent --accept-package-agreements --accept-source-agreements
        Start-Process "$env:ProgramFiles\Docker\Docker\Docker Desktop.exe" -ArgumentList "--quit"
        Start-Sleep 30
    }
}

# Git
if (-not (Get-Command git -ErrorAction SilentlyContinue)) {
    Log "📥 Installing Git..."
    winget install Git.Git --silent --accept-package-agreements
    refreshenv
}

# Python 3.10+
if (-not (Get-Command python -ErrorAction SilentlyContinue) -or (python -V) -lt "Python 3.10") {
```

```
    Log "🐍 Installing Python 3.11..."
    winget install Python.Python.3.11 --silent --accept-package-agreements
    refreshenv
}

# VS Code
if (-not (Get-Command code -ErrorAction SilentlyContinue)) {
    Log "📝 Installing Visual Studio Code..."
    winget install Microsoft.VisualStudioCode --silent --accept-package-agreements
    refreshenv
}

# CUDA 12.x (optional GPU)
Log "🎮 Checking CUDA..."
$nvsmi = "$env:ProgramFiles\NVIDIA Corporation\NVSMI\nvidia-smi.exe"
if (Test-Path $nvsmi) {
    Log "✅ NVIDIA GPU detected, CUDA assumed installed."
} else {
    Log "⚠ No NVIDIA GPU found – proceeding with CPU mode where possible."
}

# ------------------------------------------------------------
#  1. Create folder structure
# ------------------------------------------------------------
Log "📁 Creating $baseDir"
New-Item -ItemType Directory -Force $baseDir | Out-Null
Set-Location $baseDir

# ------------------------------------------------------------
#  2. Ollama (native)
# ------------------------------------------------------------
if (-not (Get-Command ollama -ErrorAction SilentlyContinue)) {
    Log "🦙 Installing Ollama..."
    winget install Ollama.Ollama --silent --accept-package-agreements
}
Log "🦙 Pulling models..."
ollama pull llama3.1:8b
ollama pull deepseek-r1:7b
Start-Process "ollama" -ArgumentList "serve" -PassThru | Out-Null
Log "🦙 Ollama listening on http://localhost:11434/v1"

# ------------------------------------------------------------
#  3. Python services (native)
# ------------------------------------------------------------
Log "🧪 Creating virtual environment..."
python -m venv venv
.\venv\Scripts\activate.ps1
Log "📦 Installing Python packages..."
pip install --upgrade pip
pip install langchain langchain-community pyautogen haystack-ai sentence-transformers faiss-cpu openai-whisper fastapi
uvicorn letta melotts
python -c "from MeloTTS.melo.download_utils import download_models; download_models()"

# ------------------------------------------------------------
#  4. Git repos (native)
# ------------------------------------------------------------
function Clone-IfNeeded {
    param($repo, $dir)
    if (-not (Test-Path "$dir\.git")) {
        Log "📥 Cloning $repo → $dir"
        git clone $repo $dir
    } else {
        Log "🔄 Pulling $dir"
        git -C $dir pull
    }
}
Clone-IfNeeded "https://github.com/oobabooga/text-generation-webui.git" "tg-webui"
Clone-IfNeeded "https://github.com/FoundationAgents/OpenManus.git"     "OpenManus"

# ------------------------------------------------------------
#  5. Text-Gen-WebUI (native)
# ------------------------------------------------------------
```

```
Start-Process "python" -WorkingDirectory "$baseDir\tg-webui" -ArgumentList ".\start_windows.bat" -PassThru | Out-Null
Log "📄 Text-Gen-WebUI → http://localhost:5000"

# ------------------------------------------------------------
#  6. Whisper FastAPI service (native)
# ------------------------------------------------------------
$whisperScript = @'
from fastapi import FastAPI, UploadFile
import whisper, tempfile, os
app = FastAPI()
model = whisper.load_model("base")
@app.post("/transcribe")
async def transcribe(file: UploadFile):
    with tempfile.NamedTemporaryFile(delete=False, suffix=".wav") as tmp:
        tmp.write(await file.read())
        result = model.transcribe(tmp.name); os.unlink(tmp.name)
    return {"text": result["text"]}
'@
$whisperScript | Out-File "$baseDir\whisper_api.py" -Encoding utf8
Start-Process "python" -ArgumentList "-m uvicorn whisper_api:app --host 0.0.0.0 --port 9000" -PassThru | Out-Null
Log "🎤 Whisper → http://localhost:9000/transcribe"

# ------------------------------------------------------------
#  7. Letta (native)
# ------------------------------------------------------------
Start-Process "python" -ArgumentList "-m letta server --model-endpoint http://localhost:11434/v1 --model llama3.1:8b" -
PassThru | Out-Null
Log "🧠 Letta → http://localhost:8283"

# ------------------------------------------------------------
#  8. MeloTTS (native)
# ------------------------------------------------------------
$ttsScript = @'
from fastapi import FastAPI
from pydantic import BaseModel
from melo.api import TTS
import torch
import soundfile as sf
import io
app = FastAPI()
class TTSRequest(BaseModel):
    text: str
    voice: str = "EN-US"
@app.post("/v1/audio/speech")
async def generate_speech(request: TTSRequest):
    device = "cuda" if torch.cuda.is_available() else "cpu"
    model = TTS(language="EN", device=device)
    speaker_ids = model.hps.data.spk2id
    speaker = request.voice if request.voice in speaker_ids else "EN-US"
    output = io.BytesIO()
    audio = model.tts_to_file(request.text, speaker_ids[speaker], None, speed=1.0)
    sf.write(output, audio, model.hps.data.sampling_rate, format="WAV")
    return {"audio": output.getvalue()}
'@
$ttsScript | Out-File "$baseDir\melo_tts_api.py" -Encoding utf8
Start-Process "python" -ArgumentList "-m uvicorn melo_tts_api:app --host 0.0.0.0 --port 8001" -PassThru | Out-Null
Log "🗣️ MeloTTS → http://localhost:8001/v1/audio/speech"

# ------------------------------------------------------------
#  9. Docker services (optional)
# ------------------------------------------------------------
if (-not $SkipDocker) {
    Clone-IfNeeded "https://github.com/hpcaitech/Open-Sora.git" "OpenSora"
    Log "🐳 Starting Docker services..."
    Set-Content "$baseDir\ai-stack-tts.yml" @'
version: "3.9"
services:
  openmanus:
    image: foundationagents/openmanus:latest
    ports: ["3000:3000"]
    environment:
      OLLAMA_URL: "http://host.docker.internal:11434"
```

```
  melo-tts:
    build:
      context: .
      dockerfile: Dockerfile
    ports: ["8001:8001"]
    deploy:
      resources:
        reservations:
          devices:
            - driver: nvidia
              count: 1
              capabilities: [gpu]
'@
    Set-Content "$baseDir\Dockerfile" @'
FROM python:3.10-slim
RUN pip install melotts torch torchaudio soundfile fastapi uvicorn
RUN python -c "from MeloTTS.melo.download_utils import download_models; download_models()"
COPY melo_tts_api.py /app/melo_tts_api.py
WORKDIR /app
CMD ["uvicorn", "melo_tts_api:app", "--host", "0.0.0.0", "--port", "8001"]
'@
    Start-Process "docker-compose" -ArgumentList "-f $baseDir\ai-stack-tts.yml up -d" -PassThru | Out-Null
    Log "🐬 Docker services up (OpenManus + MeloTTS)"
}

# ------------------------------------------------------------
# 10. Summary banner
# ------------------------------------------------------------
Log "✅ All services should now be running:"
Log "   Ollama             : http://localhost:11434/v1"
Log "   Text-Gen-WebUI     : http://localhost:5000"
Log "   Whisper            : http://localhost:9000/transcribe"
Log "   Letta              : http://localhost:8283"
Log "   MeloTTS            : http://localhost:8001/v1/audio/speech"
Log "   OpenManus          : http://localhost:3000  (if Docker)"
Log "   OpenSora           : docker exec -it open-sora bash  (if Docker)"
Log "💡 Tail logs in $baseDir\logs\"
```

🎉 END OF DOCUMENT – 11 projects, fully wired, ready to speak, with VS Code integration.