Perfect—here's an **optimized, step-by-step execution framework** tailored for your system and goals, with all the enhancements you requested:

Ultimate Windows 11 Pro AI Automation Deployment Script & SOP

For:

- **Hardware:** Threadripper Pro 7965WX, 2x RTX 5090, 512GB RAM, 4TB NVMe, 40TB+ backup
- **OS:** Windows 11 Pro, WSL2/Ubuntu 22.04
- **Project:** Local-first, multi-agent, modular, hands-off AI dev workstation (MetaGPT, Agent S, ScreenAgent, Open Computer Agent, OSCAR, Docker, VSCode, Ollama, etc.)

1. PRE-DEPLOYMENT

A. Firmware/BIOS Update (Manual Step)

- Download latest BIOS for your motherboard from Gigabyte/ASUS, copy to USB, and update BIOS first.
- Ensure "Resizable BAR," "Above 4G Decoding," and "PCIe 4.0/5.0" are enabled for RTX 5090.

B. Storage Planning

- **VHDX (Virtual Hard Disk) for AI:**
  - Mount a dedicated VHDX on a fast NVMe (e.g., `D:\AI-Dev\AI_DATA.vhdx`, 2TB+).
  - All AI models, data, and Docker/WSL volumes should point here for speed.
- **Backup:**
  - Reserve the 40TB+ for periodic backups, not primary execution.

- **Separate Project Data:**
    - If you run massive datasets, dedicate a subfolder or VHDX for “/data”.

2. BOOT DRIVE & WINDOWS INSTALL

A. Ventoy USB Prep

1. Use Ventoy to create a multi-bootable USB stick.
2. Add the following to the USB root:
    a. Windows 11 Pro ISO
    b. Latest chipset, network, and NVIDIA drivers
    c. `Autounattend.xml` (unattended Windows install config)
    d. `SetupComplete.cmd` (calls your post-install .ps1)
    e. `PostInstall.ps1` (master PowerShell script for setup below)

3. AUTOMATED POST-INSTALL SCRIPT (PostInstall.ps1)

Recommended Execution:

- Trigger from `SetupComplete.cmd` during Windows Setup (ideal, but requires scripting permissions in XML).
- **If not possible**, just run `PostInstall.ps1` manually after first boot.

[Sample PostInstall.ps1 - Modularized and Optimized]

```
# ============================
# POST-INSTALL: Ultimate AI Workstation
# ============================

# --- [ 1. Initial System Tweaks & Updates ] ---
Write-Host "Starting system tweaks, Windows updates, BIOS/Firmware
checks..."
```

```
# BIOS/Driver Reminder (manual, but log for audit)
Start-Process "notepad.exe" "D:\_BIOS_UPDATE_README.txt"

# System Update
Start-Process "ms-settings:windowsupdate"
# Optionally automate (for advanced users): Install-PackageProvider -
Name NuGet -Force; Install-Module PSWindowsUpdate -Force

# --- [ 2. Essential Drivers: Chipset, Network, NVIDIA RTX 5090 ] ---
Write-Host "Installing chipset/network/NVIDIA drivers..."
# Assumes drivers are on USB under \Drivers\
Start-Process "$env:SystemDrive\Drivers\AMD_Chipset.exe"
Start-Process "$env:SystemDrive\Drivers\LAN_Driver.exe"
Start-Process "$env:SystemDrive\Drivers\NVIDIA_RTX5090.exe"

# --- [ 3. Configure Storage: VHDX for AI ] ---
Write-Host "Mounting VHDX for AI workloads..."
# Mount AI VHDX (edit path as needed)
Mount-VHD -Path "D:\AI-Dev\AI_DATA.vhdx" -PassThru | Initialize-Disk -
PartitionStyle GPT -PassThru | New-Volume -FileSystem NTFS -
DriveLetter V

# --- [ 4. Install Core Software ] ---
$pkgs = @(
  "git", "python", "python3", "curl", "7zip", "notepad++",
"visualstudiocode", "dockerdesktop", "github-desktop"
)
foreach ($p in $pkgs) {
  winget install --silent --accept-source-agreements --accept-package-
agreements $p
}
# Confirm all GUI apps auto-start on boot if desired

# --- [ 5. VSCode Extensions (Python, Docker, Remote, Jupyter, CUDA,
etc.) ] ---
code --install-extension ms-python.python
code --install-extension ms-toolsai.jupyter
code --install-extension ms-vscode-remote.remote-wsl
code --install-extension ms-vscode-remote.remote-containers
```

```
code --install-extension ms-azuretools.vscode-docker
code --install-extension Nvidia.vscode-nvidia-cuda

# --- [ 6. Docker, WSL2, Ubuntu 22.04, NVIDIA CUDA Toolkit for WSL ] ---
wsl --install -d Ubuntu-22.04
# Or update WSL: wsl --update
# CUDA for WSL2: https://docs.nvidia.com/cuda/wsl-user-guide/index.html
wsl -d Ubuntu-22.04 -- bash -c "sudo apt update && sudo apt upgrade -y && sudo apt install -y build-essential python3-pip git docker.io nvidia-cuda-toolkit"
# Add user to docker group
wsl -d Ubuntu-22.04 -- bash -c "sudo usermod -aG docker \$USER"

# --- [ 7. Ollama & Local AI Models ] ---
Write-Host "Installing Ollama + Models..."
winget install ollama.ollama
# Optional: install with custom model dir on VHDX
# ollama --model-dir V:\OllamaModels pull llama3:70b
# (Do this for all required models, see your previous list)

# --- [ 8. Agent & Multi-Agent System Setup ] ---
Write-Host "Cloning and installing agents..."
cd V:\AI-Dev\
git clone https://github.com/simular-ai/Agent-S.git
git clone https://github.com/OthersideAI/self-operating-computer.git
git clone https://github.com/All-Hands-AI/OpenHands.git
git clone https://github.com/fosowl/agenticSeek.git
git clone https://github.com/FoundationAgents/MetaGPT.git
git clone https://github.com/coleam00/ScreenAgent.git
git clone https://github.com/OSC-AI/OSCAR.git
# ...add more as needed

# --- [ 9. Python Environment (venv, pip) for Agents ] ---
# Globally in V:\AI-Dev\venvs or per-agent in subfolders
python -m venv V:\AI-Dev\venvs\metaenv
V:\AI-Dev\venvs\metaenv\Scripts\activate
pip install -r V:\AI-Dev\MetaGPT\requirements.txt
```

```
# Repeat per agent, or centralize requirements

# --- [ 10. Docker Compose for Service Stacks ] ---
# If agent stack includes n8n, Supabase, Trivy, etc.
cd V:\AI-Dev\MicroAgentStack\
docker compose up -d

# --- [ 11. Backup/NAS Test & Mount (Optional) ] ---
# Mount or validate network share (edit as needed)
# net use Z: \\NAS\AI_Backup /user:youruser yourpass

# --- [ 12. Restore Defender, Harden System ] ---
Set-MpPreference -DisableRealtimeMonitoring $false
# Optionally install Guardz, Drata, or other compliance tools

# --- [ 13. System Cleanup and Finalize ] ---
Write-Host "System ready. Please reboot now for all settings to take
effect."

# ============================
# END SCRIPT
# ============================
```

4. NOTES & FINAL RECOMMENDATIONS

- **For truly hands-off installs:** Attach the `PostInstall.ps1` script in the `SetupComplete.cmd` (called by unattend XML during Windows install).
- **Mount all AI/data to VHDX on fast NVMe** for best performance. Models/data not in use can be symlinked or moved to backup drive.
- **Manual steps (BIOS/firmware) are necessary for optimal hardware performance—automate as much as possible, but some steps require confirmation.**
- **Validate** after first boot: Docker Desktop, VSCode, GitHub Desktop, Ollama, and all agents should launch and update normally.
- **Expand agent install/config as new agents or AI tools emerge.**

5. PROJECT FILE (FOR UPLOAD/REFERENCE)

- Copy the above as `UltimateAI-PostInstall.ps1` and place on your Ventoy USB and/or project repo.
- Reference this script/SOP in your agent stack, MetaGPT project, and onboarding docs.

This is a full, robust, future-proof framework—

You can proceed with this plan!

If you want me to generate a ready-to-upload, fully commented .ps1 and .md file for the project repo or USB, just say "export all as files".

Let me know if you want a “SetupComplete.cmd” or `Autounattend.xml` sample to complete the true zero-click experience!