Executive Framework – Automating Windows 11 Pro for AI Development & Deployment

1 Overview

This framework delivers a **hands-free, repeatable build** of a Windows 11 Pro workstation or VM ready for local-first AI research and multi-agent automation. It integrates:

- **Ventoy** multiboot USB for the install medium cite turn0search3
- A fully-attended **Autounattend.xml** combined with **SetupComplete.cmd** to pivot into PowerShell post-install cite turn0search0
- Post-install bootstrap that calls **Chocolatey** cite turn0search4 and **WinGet** import manifests cite turn0search1
- Model/runtime layer (Ollama cite turn0search2 + Python miniconda)
- Multi-modal computer-control agents (**Agent S** cite turn0search5, **Open Computer Agent** cite turn0search6, **ScreenAgent** cite turn0search7, **OSCAR** ref list cite turn0search8)
- Optional cloud-connected GitHub agents (GitHub AI coding agent) for CI tasks cite turn0news83 turn0news84

Output is a **master `Deploy-AI.ps1`** script plus supporting XML/JSON that you can place on the Ventoy stick → Windows consumes them automatically, finishing with an AI-optimised desktop in ≈30 min.

2 Prerequisites

Item	Why
64 GB USB-C SSD (exFAT)	Ventoy multiboot drive
Windows 11 Pro ISO (latest GA)	Source media
Second admin PC	To create the USB & edit XML

NVIDIA/AMD GPU (8 GB VRAM +)	Ollama GPU inference
Internet LAN/Wi-Fi	Package bootstrap & model pulls

3 Stage A – Create Bootable Media

1. **Flash Ventoy** to USB:

```
ventoy2disk.exe -i E:
```

2. Copy `Win11_Pro_x64.iso` + the following side-car files to **/ventoy/** root:
    a. `autounattend.xml` (see Appendix A)
    b. `SetupComplete.cmd` (see Appendix B)
    c. `Deploy-AI.ps1` master script
3. Add any additional ISOs (Linux rescue, firmware tools) – Ventoy menu will list them all cite turn0search3

4 Stage B – Unattended Windows 11 Install

- **Autounattend.xml** answers all Setup pages, injects local admin `aiadmin`, enables BitLocker (TPM auto-unlock), joins Workgroup, sets time-zone `Central US`, and copies the post-install scripts into `%SystemRoot%\Setup\Scripts` folder so `SetupComplete.cmd` fires automatically cite turn0search0.
- Nothing on-screen until first logon.

5 Stage C – Post-Install Bootstrap (SetupComplete.cmd)

1. Elevates PowerShell policy → Bypass.
2. Installs **Chocolatey** in silent mode `Set-ExecutionPolicy Bypass -Force; Invoke-Expression ((New-Object Net.WebClient).DownloadString('https://chocolatey.org/install.ps1'))` cite turn0search4.

3. Installs **Git**, **7-Zip**, **VS Code**, **Python 3.12**, **Miniconda**, **Docker Desktop**, **WSL2**, **NVIDIA CUDA**, etc. via **winget** import:

```
winget import -i .\winget-ai.json --accept-package-agreements --accept-source-agreements
``` cite turn0search1

4. Enables WSL feature + Ubuntu 22.04 image.
5. Reboots once (automatic).

## 6 Stage D – Model & Runtime Layer

| Component | Install Path | Notes |
|---|---|---|
| **Ollama for Windows** | `C:\Program Files\Ollama` | Native GPU build; models pulled to `C:\Users\Public\Ollama\models` cite turn0search2 |
| **MicroAgentStack** | `D:\AgentStack` | Extract from provided tar.gz, create Python venv |
| **MetaGPT** | global pip install, provides SOP agent framework | |
| **Local models (llama3, mixtral, deepseek-coder, etc.)** | via `ollama pull` list | |

A Conda env `ai-base` pins Python 3.11, PyTorch, CUDA 12.

## 7 Stage E – Agent Layer Integration

1. **Agent S** GUI-agent service → installs from GitHub release; registers as `agents` Windows service cite turn0search5.
2. **Open Computer Agent** → Node JS app; installed under `%ProgramFiles%\OCA`; service via NSSM cite turn0search6.

3. **ScreenAgent** VLM controller; uses Python; service template in Appendix C 
4. **OSCAR** (state-aware planner) added as a MicroAgentStack plugin 
5. **MetaGPT orchestration:** YAML defines roles mapping (ProductMgr, Engineer, DesktopAgent) and each local agent exposes gRPC endpoints.

## 8 Validation & Smoke Tests

1. `Invoke-AIValidation.ps1` — checks GPU driver, launches `ollama run phi4-mini` prompt, asserts response latency <200 ms.
2. GUI-agent test: Agent S opens Notepad, types `Hello AI`. ScreenAgent verifies pixel match.
3. Report JSON stored in `C:\AI-Setup\logs` and uploaded to your private GitHub Gist via PAT.

## 9 Maintenance

- **Update-All.ps1** leverages `choco upgrade all -y` + `winget upgrade --all --silent` weekly (Task Scheduler).
- Rollback: Windows System Restore point `Pre-AISetup` created before Stage C.

## 10 Execution Sequence Diagram (High-level)

```
sequenceDiagram
    participant BIOS
    participant WinPE
    participant Setup
    participant PostInstall
    participant AgentLayer
    BIOS->>WinPE: Boot ISO via Ventoy
    WinPE->>Setup: Start unattended install
```

```
    Setup->>PostInstall: Run SetupComplete.cmd
    PostInstall->>PostInstall: Deploy-AI.ps1 (packages, models)
    PostInstall->>AgentLayer: Register services
    AgentLayer-->>User: Ready for commands
```

11 Appendices

Appendix A – Autounattend.xml (excerpt)

```
<SetupConfig>
  <Run>
    <Path>SetupComplete.cmd</Path>
  </Run>
  <UserData>
    <ProductKey>XXXXX-XXXXX-XXXXX-XXXXX-XXXXX</ProductKey>
    <AcceptEula>true</AcceptEula>
  </UserData>
</SetupConfig>
```

Appendix B – SetupComplete.cmd

```
@echo off
:: Elevate & call main PS script
powershell -ExecutionPolicy Bypass -File %SystemDrive%\AI-
Setup\Deploy-AI.ps1
exit /b 0
```

Appendix C – Deploy-AI.ps1 (snippet)

```
Param([switch]$SkipModels)
# 1  Bootstrap tools
iex ((New-Object
Net.WebClient).DownloadString('https://chocolatey.org/install.ps1'))
winget import -i "$PSScriptRoot\winget-ai.json" --accept-package-
```

```
agreements --accept-source-agreements
# 2  Ollama + models
Start-Process 'C:\Program Files\Ollama\ollama.exe' -ArgumentList
'serve' -NoNewWindow
if (-not $SkipModels) {
  'llama3:70b','mixtral:8x22b','deepseek-coder:33b' | ForEach-Object
{ & "C:\Program Files\Ollama\ollama.exe" pull $_ }
}
# 3  Agent layer
pip install git+https://github.com/simular-ai/Agent-S.git
pip install git+https://github.com/niuzaisheng/ScreenAgent.git
npm install -g computer-agent
nssm install OCA "C:\Program Files\nodejs\node.exe" "C:\Program
Files\nodejs\node_modules\computer-agent\index.js"
Restart-Computer -Force
```

12 Next Steps

1. Copy this document & scripts onto your Ventoy drive.
2. Optionally tailor `winget-ai.json` for extra apps (CUDA toolkit versions, etc.).
3. Boot target machine → grab coffee → return to an AI-enabled desktop!

Appendix B – SetupComplete.cmd

```
@echo off
:: Elevate & call main PS script
powershell -ExecutionPolicy Bypass -File %SystemDrive%\AI-
Setup\Deploy-AI.ps1
exit /b 0
```

Appendix C – Deploy-AI.ps1 (snippet)

```
Param([switch]$SkipModels)
# 1  Bootstrap tools
```

```
iex ((New-Object
Net.WebClient).DownloadString('https://chocolatey.org/install.ps1'))
winget import -i "$PSScriptRoot\winget-ai.json" --accept-package-
agreements --accept-source-agreements
# 2  Ollama + models
Start-Process 'C:\Program Files\Ollama\ollama.exe' -ArgumentList
'serve' -NoNewWindow
if (-not $SkipModels) {
  'llama3:70b','mixtral:8x22b','deepseek-coder:33b' | ForEach-Object
{ & "C:\Program Files\Ollama\ollama.exe" pull $_ }
}
# 3  Agent layer
pip install git+https://github.com/simular-ai/Agent-S.git
pip install git+https://github.com/niuzaisheng/ScreenAgent.git
npm install -g computer-agent
nssm install OCA "C:\Program Files\nodejs\node.exe" "C:\Program
Files\nodejs\node_modules\computer-agent\index.js"
Restart-Computer -Force
```

12 Next Steps

1. Copy this document & scripts onto your Ventoy drive.
2. Optionally tailor `winget-ai.json` for extra apps (CUDA toolkit versions, etc.).
3. Boot target machine → grab coffee → return to an AI-enabled desktop!