CUTAWAY SECURITY
INFOSEC CONSULTANTS

Framework Desktop AI Workstation Installation Guide

Building a Local LLM Development Server for Cybersecurity Research

Version 6 — February 2026

Cutaway Security, LLC
https://www.cutawaysecurity.com
https://github.com/cutaway-security

Table of Contents

1. Hardware Overview

This guide covers building an AI development workstation using the Framework Desktop with AMD's Ryzen AI Max+ 395 processor. The system is optimized for running large language models (LLMs) locally without cloud dependencies.

Component	Selection
System	AMD Ryzen AI Max+ 395 — 128GB LPDDR5x-8000
Storage 1	WD_BLACK SN850X NVMe M.2 2280 — 4TB
Storage 2	WD_BLACK SN850X NVMe M.2 2280 — 4TB
CPU Fan	Noctua NF-A12x25 HS-PWM

Platform Specifications:
• 16 cores / 32 threads (Zen 5 architecture)
• Radeon 8060S integrated GPU (40 RDNA 3.5 compute units)
• 128GB unified LPDDR5x-8000 memory (~125 Gi usable)
• ~256 GB/s theoretical bandwidth (212–215 GB/s real-world)
• 120W sustained / 140W boost TDP
• Connectivity: 2x USB4, 2x DisplayPort, HDMI, 5GbE, Wi-Fi 7

Note: The unified memory architecture allows the GPU to access up to 96GB of system memory for model inference, enabling 70B+ parameter models to run locally.

2. AI-Assisted Planning

This build was planned and troubleshot with assistance from Claude, Anthropic's AI assistant. The collaboration spanned hardware selection, partition layouts, GPU memory configuration, and debugging issues like LUKS boot prompts and API changes in third-party libraries. One day of the effort included installing and configuring the supporting NUC Proxmox cluster.

Any capable AI assistant can fill this role — ChatGPT, Gemini, or others. The key benefit is having a knowledgeable collaborator available to work through technical decisions, generate configuration files, and troubleshoot errors in real-time. For complex builds like this one, AI assistance reduces the time spent searching documentation and forums, letting you focus on the actual work.

Future development efforts on this platform will continue using AI collaboration. Tools and improvements will be shared through the CutSec GitHub repositories at https://github.com/cutaway-security.

3. Operating System Selection

Kubuntu 24.04 LTS

• Ships with kernel 6.17 and Mesa 25.2 (HWE stack) — full Strix Halo support
• LTS support until April 2029
• Desktop installations track the HWE stack automatically
• Clean upgrade path to Kubuntu 26.04 LTS

Install Media: Kubuntu 24.04.3 or later ISO. After installation, run `sudo apt update && sudo apt full-upgrade` to pull the HWE stack.

4. Storage Layout

Both drives use GPT partition tables. Sensitive partitions use LUKS encryption.

Drive 1 (4TB) — System + Development:

#	Size	Mount	Encrypt	Purpose
1	1 GB	/boot/efi	No	UEFI firmware
2	2 GB	/boot	No	Kernel, initramfs, GRUB
3	200 GB	/	Yes	OS, packages, drivers
4	500 GB	/var/lib/docker	Yes	Docker images, Qdrant indexes
5	16 GB	swap	Yes	Encrypted swap
6	~2.9 TB	/home	Yes	Projects, API keys, SSH keys

Drive 2 (4TB) — AI Models + Datasets:

#	Size	Mount	Encrypt	Purpose
1	1 TB	/data/datasets	Yes	PCAPs, logs, vendor docs
2	~2.7 TB	/data/models	No	Public LLM model files

Note: The /data/models partition is unencrypted because it contains only publicly downloadable LLM files. This avoids decryption overhead during model loading.

5. Phase 1: Hardware Assembly

Follow Framework's official assembly instructions. Key points:

- ■ Install both NVMe drives in M.2 slots
- ■ Connect Noctua fan to CPU fan header
- ■ Verify all connections before closing chassis
- ■ Connect power, display, and peripherals

6. Phase 2: BIOS Configuration

The AMD Ryzen AI Max+ 395 platform works with BIOS defaults. No special virtualization settings (SVM, IOMMU) are required unless running local VMs.

- ■ Enter BIOS setup (press DEL during boot)
- ■ Verify both NVMe drives are detected
- ■ Optionally: Set UMA Frame Buffer to 512MB (GTT handles GPU allocation)
- ■ Save and exit

7. Phase 3: OS Installation

Create Installation Media:

Download Kubuntu 24.04.3+ ISO and write to USB using Balena Etcher or dd.

Boot and Install:

- ■ Boot from USB (F12 for boot menu)
- ■ Select 'Install Kubuntu'
- ■ Choose language and keyboard layout
- ■ Select 'Manual partitioning'
- ■ Create partitions per the storage layout (enable encryption for marked partitions)
- ■ Set timezone and create user account
- ■ Complete installation and reboot

First Boot:

You will be prompted for LUKS passphrases. Enter your passphrase for each encrypted partition (or configure keyfile auto-unlock — see Appendix A).

Post-Install Updates:

```
sudo apt update && sudo apt full-upgrade -y
sudo reboot
```

Verify Installation:

```
uname -r                    # Expect: 6.17.x or higher
glxinfo | grep "OpenGL"     # Expect: Mesa 25.2.x
free -h                     # Expect: ~125 Gi total
```

8. Phase 4: GPU Memory Configuration

Configure the GPU Translation Table (GTT) to allow the GPU to use up to 96GB of system memory for model inference.

Set GTT Size:

```
sudo nano /etc/default/grub

# Add to GRUB_CMDLINE_LINUX_DEFAULT:
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash amdgpu.gttsize=98304"

# Update GRUB and reboot:
sudo update-grub
sudo reboot
```

Verify GPU Configuration:

```
# Check GPU driver
lspci -k | grep -EA3 'VGA|3D|Display'
# Expect: Kernel driver in use: amdgpu

# Check Vulkan
vulkaninfo --summary
# Expect: Radeon 8060S Graphics (RADV GFX1151)

# Check GTT allocation
cat /sys/class/drm/card*/device/mem_info_gtt_total | head -1 | \
    awk '{printf "GTT: %.1f GB\n", $1/1024/1024/1024}'
# Expect: GTT: 96.0 GB
```

Device Permissions:

```
# Add user to video and render groups
sudo usermod -aG video,render $USER

# Log out and back in, then verify:
groups $USER
# Expect: video render (among others)
```

9. Phase 5: Core Tools Installation

Essential Packages:

```
sudo apt install -y build-essential git curl wget vim htop tmux \
    python3 python3-pip python3-venv python3.12-venv \
    unzip jq tree net-tools tshark

# GPU monitoring (nvtop works better than radeontop for gfx1151)
sudo apt install -y nvtop vulkan-tools mesa-utils
```

Credential Management (pass):

Use `pass` for GPG-encrypted credential storage that works over SSH.

```
sudo apt install -y pass pinentry-curses

# Generate GPG key (if needed)
gpg --gen-key

# Initialize pass
pass init "your-email@example.com"

# Configure GPG agent for SSH sessions
echo 'export GPG_TTY=$(tty)' >> ~/.bashrc

mkdir -p ~/.gnupg
cat >> ~/.gnupg/gpg-agent.conf << 'EOF'
default-cache-ttl 28800
max-cache-ttl 28800
pinentry-program /usr/bin/pinentry-curses
EOF

gpg-connect-agent reloadagent /bye
```

Security Tools:

```
sudo apt install -y nmap netcat-openbsd dnsutils whois \
    tcpdump wireshark-common

# Python security packages
sudo apt install -y python3-scapy python3-ipython
pip install pwntools pyshark --break-system-packages

# SecLists wordlists
sudo git clone --depth 1 \
    https://github.com/danielmiessler/SecLists.git /usr/share/seclists
```

Node.js (for Claude Code):

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.40.3/install.sh | bash
source ~/.bashrc
nvm install --lts
nvm use --lts
```

Docker:

```
sudo apt install -y docker.io docker-compose-v2
sudo usermod -aG docker $USER
newgrp docker
docker run hello-world
```

Tailscale:

```
curl -fsSL https://tailscale.com/install.sh | sh
sudo tailscale up
tailscale status
```

Remote Desktop (xrdp):

```
sudo apt install -y xrdp
sudo systemctl enable xrdp
sudo systemctl start xrdp
sudo usermod -aG ssl-cert xrdp
sudo systemctl restart xrdp
```

Note: RDP is accessible over Tailscale using the machine's tailnet hostname on port 3389. No additional firewall rules needed.

10. Phase 6: LLM Stack Setup

Ollama provides the primary interface for model management and inference. It wraps llama.cpp with automatic model downloads and a REST API.

Install Ollama:

```
curl -fsSL https://ollama.com/install.sh | sh
```

Configure for Docker Access:

```
sudo systemctl edit ollama

# Add between the comments:
[Service]
Environment="OLLAMA_HOST=0.0.0.0"

# Save and restart:
sudo systemctl daemon-reload
sudo systemctl restart ollama
```

Download and Test Models:

```
# Test model
ollama pull llama3.2
ollama run llama3.2 "Hello, what is the capital of France?"

# Embedding model for RAG
ollama pull nomic-embed-text

# Production models (optional)
ollama pull qwen2.5-coder:32b
ollama pull llama3.3:70b-instruct-q4_K_M
```

Open WebUI:

```
docker run -d -p 3000:8080 \
    --add-host=host.docker.internal:host-gateway \
    -v open-webui:/app/backend/data \
    --name open-webui \
    --restart always \
    ghcr.io/open-webui/open-webui:main

# Access at http://localhost:3000
```

Expected Performance:

Model Size	Speed	Notes
7-8B	~50+ tok/s	Fast, responsive
30-32B	~15-20 tok/s	Good for coding
70B (Q4-Q6)	~5 tok/s	Workable, noticeable latency

11. Phase 7: Claude Code

Claude Code is Anthropic's command-line tool for AI-assisted development.

Install Claude Code:

```
npm install -g @anthropic-ai/claude-code
```

Store API Key Securely:

```
# Store key with pass
pass insert api/anthropic
# Enter your Anthropic API key when prompted

# Configure shell to load key
echo 'export ANTHROPIC_API_KEY=$(pass api/anthropic 2>/dev/null)' >> ~/.bashrc
source ~/.bashrc
```

Test:

```
mkdir ~/Projects/test && cd ~/Projects/test
claude
```

Note: After reboot, the first use will prompt for your GPG passphrase. It's cached for 8 hours by gpg-agent.

12. Phase 8: RAG Pipeline Setup

Set up a Retrieval-Augmented Generation pipeline using Qdrant vector database and local embeddings via Ollama.

Deploy Qdrant:

```
docker run -d -p 6333:6333 -p 6334:6334 \
    -v qdrant-data:/qdrant/storage \
    --name qdrant \
    --restart always \
    qdrant/qdrant:latest

# Verify
curl http://localhost:6333/healthz
```

Create RAG Environment:

```
python3 -m venv ~/venvs/rag
source ~/venvs/rag/bin/activate
pip install --upgrade pip

pip install langchain langchain-community langchain-qdrant \
    qdrant-client sentence-transformers \
    pymupdf python-docx scapy pyshark \
    tqdm rich ollama
```

Test Embedding Generation:

```
curl http://localhost:11434/api/embeddings -d '{
  "model": "nomic-embed-text",
  "prompt": "Modbus TCP function code 3 read holding registers"
}'
# Expect: JSON response with embedding array
```

Dataset Directory Structure:

```
sudo mkdir -p /data/datasets/{pcaps,syslogs,vendor-docs,protocol-captures}
sudo chown -R $USER:$USER /data/datasets
```

13. Appendix A: LUKS Multi-Partition Boot Troubleshooting

Problem:

When booting with multiple LUKS-encrypted partitions, Plymouth (graphical boot splash) may fail to display passphrase prompts for secondary partitions. The screen goes black and keyboard appears unresponsive.

Solution: Keyfile Auto-Unlock

Create a keyfile on the root partition that automatically unlocks other partitions. Only one passphrase is needed at boot (for root).

Step 1: Disable Plymouth

```
sudo nano /etc/default/grub
# Remove 'quiet splash' from GRUB_CMDLINE_LINUX_DEFAULT

sudo update-grub
sudo reboot
```

Step 2: Create Keyfile

```
sudo dd if=/dev/urandom of=/crypto_keyfile.bin bs=4096 count=1
sudo chmod 400 /crypto_keyfile.bin
```

Step 3: Add Keyfile to Encrypted Partitions

```
# Identify encrypted partitions
lsblk -f | grep crypto

# Add keyfile to each (except root)
sudo cryptsetup luksAddKey /dev/nvmeXnYpZ /crypto_keyfile.bin
# Repeat for each encrypted partition
```

Step 4: Update Initramfs

```
sudo update-initramfs -u -k all
sudo reboot
```

Note: You may see warnings during boot ('could not locate crypto_keyfile.bin'). This is a timing issue — be patient. Boot completes successfully after 1-2 minutes.

14. Appendix B: Backup and Maintenance

Backup Script:

```
mkdir -p ~/scripts ~/backups
cat > ~/scripts/backup-configs.sh << 'EOF'
#!/bin/bash
BACKUP_DIR="$HOME/backups/configs-$(date +%Y%m%d)"
```

```
mkdir -p "$BACKUP_DIR"
cp /etc/default/grub "$BACKUP_DIR/"
cp -r /etc/udev/rules.d/ "$BACKUP_DIR/"
cp -r /etc/docker/ "$BACKUP_DIR/"
cp ~/.bashrc "$BACKUP_DIR/"
dpkg --get-selections > "$BACKUP_DIR/installed-packages.txt"
echo "Backup completed: $BACKUP_DIR"
EOF
chmod +x ~/scripts/backup-configs.sh

# Run backup
~/scripts/backup-configs.sh
```

Firmware Updates:

```
sudo fwupdmgr refresh
sudo fwupdmgr get-updates
sudo fwupdmgr update  # If updates available
```

Ongoing Maintenance:

- System updates: `sudo apt update && sudo apt upgrade`
- Docker cleanup: `docker system prune -a` (periodically)
- Monitor disk usage: `df -h`
- Monitor GPU health: `nvtop`
- Monitor Qdrant: `curl http://localhost:6333/healthz`

—

Go forth and do good things,
Don C. Weber
Cutaway Security, LLC