WORKFLOW SYSTEMS • 2026 INFRASTRUCTURE

How to Install Docker on Windows and Mac

Modern dev environments run on containerized infrastructure: AI agents, local LLMs, vector databases, and automation stacks. This technical blueprint guides you through configuring Docker Desktop, WSL2, and multi-container orchestration.

`Docker Desktop` `WSL2` `Docker Compose` `AI Infrastructure` `DevOps` `2026 Workflow`

The 2026 Shift: Production workflows now demand local isolated layers for data and models. Before initiating complex configurations (e.g., local multi-agent environments), a pristine, underlying container runtime layer is required.

THE STANDARD CORE STACK

The standard stack combination ensures native execution speeds, reliable volumes, and flawless toolchain cross-talk.

STACK OPTION	TARGET PLATFORM	STRATEGIC ENGINEERING RECOMMENDATION
Docker Desktop + WSL2 + VS Code	Windows Developers	**Best overall Windows workflow.** Integrates Ubuntu file system perfectly with native host automation loops.
Docker Desktop + Compose	Mac Developers	**Best native Mac workflow.** Production standard for stable cross-compilation on Apple Silicon.
OrbStack + Compose	Advanced Mac Users	Lightweight alternative specializing in maximum performance and low memory overhead.
Rancher Desktop + Compose	Enterprise Teams	Open-source focused architecture optimized for Kubernetes-centric delivery structures.

Recommended Hardware Thresholds

16GB – 32GB	NVMe SSD	Apple M1–M4	Windows 11	≥ 40 GB
RAM PREFERRED	STORAGE CLASS	MAC ARCHITECTURE	OS BASELINE	FREE SPACE

WINDOWS INSTALLATION

01 Enable Hardware Virtualization

Ensure Intel VT-x or AMD-V is enabled inside your motherboard system BIOS configuration. Docker requires direct processor-level virtualization to execute its internal hypervisor layer.

MACOS INSTALLATION

01 Download Targeted Installer

Acquire the correct installer compilation explicitly tailored for your CPU architecture:

- **Apple Silicon:** M1/M2/M3/M4 ARM64 SoC
- **Intel Chip:** Legacy x86 Core processors

02 Deploy WSL2 Kernel

Open PowerShell as Administrator and run the environment bootstrapper command:

```
wsl --install
```
CLI

Restart your workstation immediately following execution. Verify status via:

```
wsl --status
```
CLI

03 Provision Ubuntu Distribution

Install Ubuntu via the Microsoft Store. This sets up your dedicated Linux dev subsystem, optimizing network proxy links and directory routing structures.

04 Mount Docker Desktop Client

Download the Windows installer from the official portal. Execute the package and strictly verify that **"Use the WSL 2 based engine"** is ticked during configuration prompts.

05 Map Cross-Distro Integration

Navigate to *Settings* → *Resources* → *WSL Integration* inside Docker Desktop. Toggle the slider to activate integration on your **Ubuntu** system, then click Apply.

02 Mount and Authorize

Open the downloaded `.dmg` archive file. Drag the Docker application directly into your local `/Applications` directory path. Launch it from Launchpad.

03 Delegate System Privileges

Upon initial launch, macOS Ventura/Sonoma/Sequoia will prompt for administrative access to configure privileged networking helpers and virtual link elements. Grant authorization.

04 Initialize Daemon Runtime

Monitor the visual status indicator icon residing inside the top system menu bar. Wait until the whale logo transitions to a steady state showing running status.

05 Verify Apple Silicon Execution

Modern ARM setups run seamlessly. Docker Desktop leverages advanced internal Rosetta 2 frameworks natively to translate legacy x86-only container payloads on demand.

VERIFICATION & RUNTIME ASSESSMENT

A successful environment layout requires validating structural system communication. Run the verification stack inside your active terminal:

```
# Step 1: Validate CLI Availability
docker --version

# Step 2: Test End-to-End Image Pull, Daemon Routing, and Container Lifecycle
docker run hello-world
```
CLI

✓ If the success payload prints to your shell, your container environment is fully provisioned.

ADVANCED AI INFRASTRUCTURE STACK ORCHESTRATION

Single container execution is foundational, but real world workflows (like multi-component AI stacks consisting of UI frontends, local LLM engines, and vector store layers) rely on **Docker Compose**. This framework constructs complete local microservice meshes from a unified configuration map.

$$Stack\_State(Services, Volumes, Networks) \longrightarrow ext{docker compose up -d}$$

Essential Multi-Service Stack Lifecycle Commands:

```cli
# Build, create, and launch the entire defined infrastructure mesh silently
docker compose up -d

# View live aggregate standard output logs across active stack nodes
docker compose logs -f

# Destruct services, clear internal bridge routing loops, maintain safe volumes
docker compose down
```

Typical 2026 AI Developer Stack Composition:

- **Open WebUI / Frontend** (User Portal)
- **Ollama / vLLM** (Local Inference Node)
- **Qdrant / Milvus** (Vector Data Store)
- **PostgreSQL** (Relational State Store)

RECOMMENDED PROFESSIONAL TOOLCHAIN EXTENSIONS

- **Official VS Code Docker Extension:** Provides absolute management over local registries, active nodes, log streaming, and container file systems without changing windows.
- **Dev Containers (Microsoft):** Allows your primary code editor instance to live *inside* the active container context, ensuring absolute development parity across remote team nodes.
- **YAML Language Support:** Eliminates spacing compilation errors inside your Docker Compose descriptors through strict syntax highlighting and dynamic inline schema verification.

PRODUCTION TROUBLESHOOTING MATRIX

Error: Docker Daemon Connection Failure / Engine Timeout

Resolution: On Windows, re-verify your terminal state by entering `wsl --status` to ensure the layer is running. On Mac, check if the system menu bar whale icon is green. Restarting Docker Desktop cleans orphaned socket pipes.

Error: Hardware Virtualization Blocked or Disabled (BIOS)

Resolution: Reboot into your machine's system firmware setup screens. Enable "Intel Virtualization Technology" or "SVM Mode" underneath your CPU parameters configuration pane.

Error: Permission Denied Exceptions inside WSL2 Ubuntu Terminal

Resolution: Ensure your Linux system user account belongs to the native docker system group: `sudo usermod -aG docker $USER`. Log out and log back in to reload your profile permissions.

Architecture Discrepancy: "no matching manifest for linux/arm64/v8"

Resolution: Force the platform emulation flag inside your local launch config or prepend your CLI execution string: `docker run --platform linux/amd64 image_name` to leverage Rosetta translation layers natively.