ODYSSEUS SETUP
STARTER PACK

20-page SOP: 30 modules, 300 detailed checks

Official-source first. One route at a time. New versioned PDF path avoids stale 4-page browser caches.

01 Start with the official scope

01 Use this pack for first install, first login, model connection, safe support handoff, and basic recovery only.
02 Do not treat the pack as a paid license for Odysseus; the app itself comes from the public GitHub project.
03 Keep the free project boundary clear when asking for help: you paid for setup guidance, not private source code.
04 Do every step on the computer that will actually run Odysseus, not on a phone or unrelated laptop.
05 Use Google Chrome or another full desktop browser so local URLs, downloads, and security warnings are visible.
06 Keep one notebook line for the operating system, install route, local URL, port, and model provider you chose.
07 Do not open five tutorials at once; one active instruction source prevents mixed Windows, macOS, Linux, and Docker steps.
08 If the official README changed after this PDF was generated, follow the README for commands and use this PDF for checks.
09 Stop immediately if any guide asks for API keys, admin passwords, cookies, seed phrases, or remote-control access in comments.
10 When stuck, collect facts first: exact command, first error line, system version, install route, and what changed right before failure.

02 Verify the official source

01 Open github.com/pewdiepie-archdaemon/odysseus directly in the browser address bar, not through a shortened link.
02 Confirm the owner text is exactly pewdiepie-archdaemon and the repository name is exactly odysseus before cloning.
03 Prefer the README commands from GitHub because old videos may show ports, scripts, or settings that have changed.
04 Use the green Code button only if you understand whether you are cloning with Git or downloading a ZIP snapshot.
05 Avoid reuploaded EXE installers, APKs, ZIP mirrors, Discord file drops, and comment commands from unknown accounts.
06 Check that the clone command starts with https://github.com/pewdiepie-archdaemon/odysseus.git before you paste it.
07 If a copied command includes curl, powershell download, base64, or a shortened URL, pause and verify every URL first.
08 Do not enter secrets into GitHub issues, YouTube comments, Reddit posts, or public troubleshooting screenshots.
09 If GitHub is blocked or unavailable, wait or use a trusted network instead of downloading from an unverified mirror.
10 Save the repository URL in your notes so future updates and support requests refer to the same source.

Sources: official Odysseus README, Docker, Python, Ollama, Tailscale, and Cloudflare documentation.

03 Choose the right install route

01 Use Docker when you want cleaner isolation and the README says Docker is recommended for your path.
02 Use native Windows when you want the official PowerShell launcher route and do not need Linux GPU serving.
03 Use native macOS on Apple Silicon when you care about local Metal acceleration, because Docker Desktop cannot use Metal GPU.
04 Use Linux or WSL2 for advanced CUDA or ROCm model-serving routes when local GPU serving is the real goal.
05 Do not run Docker setup and native setup in the same folder at the same time; mixed artifacts make errors harder to read.
06 If you are non-technical, pick the route with the fewest commands that matches your operating system exactly.
07 If you already have Docker Desktop installed and working, Docker is usually the fastest controlled first attempt.
08 If Docker Desktop will not start, do not debug Odysseus yet; fix Docker or switch to the native route first.
09 If Python dependency errors appear on native setup, do not randomly install packages globally; rebuild the venv route cleanly.
10 Record the route name in your notes as Docker, native Windows, native macOS, native Linux, or WSL2.

04 Prepare Windows before installing

01 Open Settings and confirm Windows 10 or Windows 11 is fully updated before installing Docker, Python, or Ollama.
02 Use PowerShell for the official Windows launcher, and run commands from the odysseus folder only.
03 Install Git for Windows if git --version is not recognized after reopening PowerShell.
04 Install Python 3.11 or newer if you use the native Windows route; older Python versions can fail dependency resolution.
05 Use py -3.11 or py -3.12 when python points to the wrong interpreter or an old Microsoft Store alias.
06 Create a simple path such as C:\Odysseus and avoid OneDrive, Dropbox, spaces, emojis, and non-English folder names for first setup.
07 If Activate.ps1 is blocked, use the current-user execution policy fix from Python guidance, not a system-wide security disable.
08 Restart PowerShell after installing Git, Python, Docker, or Ollama so PATH updates are visible.
09 Keep Windows Security enabled; do not disable antivirus globally just because a dependency install is slow.
10 For Windows local models, start with Ollama unless you intentionally set up Linux, WSL2, CUDA, and a separate serving stack.

05 Prepare macOS before installing

01 Confirm whether the Mac is Apple Silicon or Intel from About This Mac before choosing Docker or native setup.
02 For Apple Silicon GPU use, prefer the native macOS script because Docker on macOS does not expose Metal GPU acceleration.
03 Use Terminal from Applications or Spotlight and keep all commands inside the odysseus project folder.
04 Install command-line developer tools only when Git or build dependencies ask for them, then rerun the failed command.
05 If the README says the macOS script opens port 7860, open that exact port instead of assuming port 7000.
06 Do not mix Homebrew Python, pyenv Python, and system Python unless you can explain which interpreter the venv uses.
07 If a permission prompt appears for network, documents, or developer tools, read it before approving.
08 Keep the app bound to localhost until login and model connection work on the Mac itself.
09 For phone access from a Mac, finish desktop setup first, then decide between trusted LAN, Tailscale, or Cloudflare Access.
10 If AirPlay or another service holds port 7000, use the README's macOS port or set a clear APP_PORT value.

06 Prepare Linux or WSL2 safely

01 Use Linux or WSL2 when you need a Linux-native Docker, CUDA, ROCm, vLLM, SGLang, or server-style environment.
02 On Windows with Docker Desktop, enable the WSL2 backend from Docker settings before expecting Linux containers to work.
03 Check wsl.exe -l -v if Docker or Linux commands behave as if they are running in the wrong distribution.
04 Keep the project inside the Linux filesystem for WSL2 development, not deep inside a Windows-synced cloud folder.
05 Install Python 3.11+, Git, and system build tools from the distribution package manager when native Linux setup needs them.
06 Do not install Docker Engine inside WSL if Docker Desktop already owns the Docker integration; duplicate engines cause confusion.
07 Check GPU visibility with the runtime's own tool before blaming Odysseus for slow model loading.
08 Run one shell session for setup and keep the terminal open while dependencies build or models download.
09 Avoid running setup as root unless a package manager explicitly requires sudo; project files should belong to your user.
10 If WSL memory grows after heavy Docker builds, close unused containers and reclaim resources before retesting.

Use the official README for commands when it changes; use this PDF for the checklist discipline.

07 Clone and organize the project

01 Create one parent folder named OdysseusSetup, AIApps, or another simple name you can find tomorrow.
02 Open the terminal in that parent folder before running git clone so the odysseus directory lands in the right place.
03 Run git clone https://github.com/pewdiepie-archdaemon/odysseus.git exactly once for a fresh install.
04 After cloning, run cd odysseus and confirm the README, setup scripts, and requirements files are visible.
05 If the odysseus folder already exists, decide whether you are updating, repairing, or starting over before overwriting anything.
06 Do not manually drag random files into the project folder; rebuild from Git and official scripts when possible.
07 Keep .env, local database files, and generated passwords private; never attach the whole folder to an email.
08 If using ZIP download instead of Git, understand that updates will require a new ZIP and manual file replacement.
09 Use git status after changes only if you understand Git; non-developers can simply keep setup notes instead.
10 Before major repair, copy only your private configuration notes and data backups, not broken venv or container artifacts.

08 Run the Docker route

01 Open Docker Desktop first and wait until the engine is running before typing docker compose commands.
02 From the odysseus folder, copy .env.example to .env if the README recommends explicit defaults.
03 Run docker compose up -d --build from the project root, not from your home directory or Downloads folder.
04 Watch for image build failures, missing files, port conflicts, and permission errors before opening the browser.
05 Use docker compose ps to confirm containers are running or exited before guessing from the browser alone.
06 Open http://localhost:7000 only after the container is healthy or logs show the server has started.
07 If port 7000 is busy, set APP_PORT=7001 in .env and recreate the container instead of editing random code files.
08 Keep APP_BIND at 127.0.0.1 until desktop login works; use 0.0.0.0 only for intentional LAN or proxy access.
09 For Docker-to-host Ollama, use the README's host.docker.internal endpoint rather than plain localhost inside the container.
10 Stop Docker cleanly with docker compose down when you need a controlled restart or config reload.

Use the official README for commands when it changes; use this PDF for the checklist discipline.

09 Run the native Windows route

01 Open PowerShell, cd into the odysseus folder, and keep that window open for the full first launch.
02 Run powershell -ExecutionPolicy Bypass -File .\launch-windows.ps1 only from the official repository folder.
03 Let the launcher create the venv, install dependencies, run setup, and start the server before clicking around.
04 If the launcher stops, copy the first meaningful error line, not the final repeated retry or stack trace tail.
05 If doing it manually, create the venv with py -3.11 -m venv venv or another installed 3.11+ launcher version.
06 Activate with venv\Scripts\Activate.ps1 only after the venv exists and PowerShell is in the project folder.
07 Install requirements inside the venv with pip install -r requirements.txt; avoid global pip installs.
08 Run python setup.py before uvicorn so generated configuration and first-run setup are present.
09 Start uvicorn on 127.0.0.1 and port 7000 unless you intentionally changed APP_PORT or APP_BIND.
10 After the browser opens successfully, save the exact start command and folder path for tomorrow.

10 Run native macOS or Linux

01 Open Terminal and cd into the odysseus folder before creating a virtual environment or running scripts.
02 Create the venv with python3 -m venv venv when using the native Linux or generic macOS path.
03 Activate with source venv/bin/activate and confirm the prompt or which python points into the venv.
04 Install dependencies with pip install -r requirements.txt while the venv is active.
05 Run python setup.py before starting uvicorn so first-run setup can generate expected local files.
06 Start with python -m uvicorn app:app --host 127.0.0.1 --port 7000 unless the README gives a platform-specific script.
07 On Apple Silicon, use ./start-macos.sh when the README recommends it for Metal-friendly local model routes.
08 If the macOS script uses port 7860, open http://127.0.0.1:7860 instead of localhost:7000.
09 Install tmux only when Cookbook background downloads or serves require it; do not install optional tools blindly.
10 Use Ctrl+C to stop native uvicorn, then rerun the same command after changing settings.

11 Find the first working browser page

01 Open the exact URL printed by the terminal; do not guess a different port because another tutorial used it.
02 Use http://localhost:7000, http://127.0.0.1:7000, or the README's macOS port depending on route.
03 If the page does not load, check whether the terminal is still running and whether it printed a startup error.
04 If Chrome says connection refused, the server is not listening on that port yet; refreshes will not fix it.
05 If Chrome shows a different app, another service owns the port; stop it or change APP_PORT and restart Odysseus.
06 If Docker runs but localhost fails, compare docker compose ps, docker compose logs, and the published port.
07 If native setup runs but localhost fails, confirm uvicorn host and port match the browser address.
08 Do not expose to LAN or public internet before you can log in from the same machine.
09 When a page appears, take a private note of the URL, install route, and startup command.
10 If login is impossible, inspect the terminal output for the generated admin password before resetting anything.

12 Handle admin password and login

01 Look for the generated admin password or first-login instruction in the terminal output during first setup.
02 Copy the password into a private password manager or local note that is not synced to a public workspace.
03 Do not email the admin password to support; support can guide where to find it without seeing it.
04 If you lose the password, search the official README and logs before deleting data or reinstalling the app.
05 Confirm AUTH_ENABLED stays true before binding the app outside localhost or placing it behind a proxy.
06 Use a long unique admin password if the app asks you to replace a temporary password.
07 Avoid browser auto-fill confusion by logging in with one browser profile during first setup.
08 After login, confirm you can reach Settings before configuring models, APIs, tools, or external access.
09 If login loops, capture browser URL, server logs, and whether cookies are blocked for localhost.
10 Treat any screenshot of the logged-in dashboard as private if it shows documents, emails, calendars, keys, or local paths.

13 Install and test Ollama

01 Install Ollama from the official Ollama download page or documentation, not from mirrored installers.
02 On Windows, the normal installer runs in the user account and does not require administrator rights.
03 After installation, check that the Ollama tray app or background service is running before connecting Odysseus.
04 Open http://localhost:11434 or run an ollama command to confirm the local server responds.
05 Pull one small model first so you prove the model path before downloading a large multi-gigabyte model.
06 Use ollama list to see models already present and avoid downloading duplicates by accident.
07 If model storage is too small, move OLLAMA_MODELS before downloading huge models, not after the disk is full.
08 On Windows, use the documented local log folders when Ollama fails to launch or update.
09 If Ollama is inside Docker or another machine, write down that address; do not assume localhost means the same thing everywhere.
10 Do not start performance tuning until one small model can answer a short prompt reliably.

14 Connect Odysseus to a model

01 Confirm Odysseus login works before changing model settings; model errors should not hide app startup errors.
02 For native Odysseus and host Ollama, start with an endpoint like http://localhost:11434/v1 when the README supports it.
03 For Docker Odysseus and host Ollama, use http://host.docker.internal:11434/v1 as documented by the README.
04 If Ollama must listen beyond loopback for Docker, set OLLAMA_HOST intentionally and understand the network exposure.
05 For external APIs, paste keys only into the official settings screen or private .env file described by the app.
06 Name the provider clearly in notes: Ollama local, OpenAI-compatible API, OpenRouter, Groq, Anthropic, or another supported route.
07 Start with one provider and one model; adding multiple providers before a first chat works creates false leads.
08 Send a one-word prompt first, then a short practical prompt, before using documents, tools, or agents.
09 If a model appears but replies fail, check endpoint, model name, API key, and first server error in that order.
10 If responses are slow, separate model performance from Odysseus setup by testing the same model directly in Ollama or the provider console.

Use the official README for commands when it changes; use this PDF for the checklist discipline.

15 Use external API providers safely

01 Use an external API provider when local hardware is weak, disk space is limited, or the user needs reliability more than local privacy.
02 Create the API key inside the provider dashboard yourself; do not ask support to create, store, or receive the key.
03 Paste the key only into Odysseus settings or a private .env field documented by the app, never into email or chat.
04 Set a low spending limit or prepaid balance before testing so a bad loop cannot create a surprise bill.
05 Confirm the provider supports the model name and endpoint format you choose before blaming Odysseus.
06 For OpenAI-compatible providers, check base URL, API key, model name, and streaming support as four separate settings.
07 If a provider returns unauthorized, rotate the key only after verifying it was pasted into the right field without spaces.
08 If a provider returns rate limit or quota errors, fix billing, quota, or model tier before changing local setup.
09 Never put a production company API key into a beginner test install; use a restricted key or a fresh test project.
10 Document the provider name and model in the handoff notes without writing the secret value.

16 Use environment settings carefully

01 Treat .env as a private local configuration file and never upload it, paste it publicly, or email it.
02 Copy .env.example to .env only when the README recommends explicit defaults for the route you use.
03 Change one setting at a time and restart the app so you can connect cause and effect.
04 Use APP_PORT only for port conflicts; changing ports will not fix model, login, or Python dependency errors.
05 Use APP_BIND=0.0.0.0 only for intentional LAN, VPN, or reverse-proxy access after authentication works.
06 Keep AUTH_ENABLED=true whenever anyone besides the local user might reach the app.
07 Put provider API keys into the setting name the app expects; do not invent variable names from other projects.
08 After changing .env for Docker, recreate or restart the container so the new environment is loaded.
09 After changing .env for native setup, stop and restart uvicorn or the platform script.
10 If a required setting is missing, fail loudly, fix the missing value, and rerun instead of hiding the warning.

Use the official README for commands when it changes; use this PDF for the checklist discipline.

17 Read logs without panic

01 The first real error line is usually more useful than the final line after many retries.
02 For Docker, run docker compose logs --tail=120 odysseus first so the output is bounded and readable.
03 For Docker status, run docker compose ps before reading logs to see whether a container exited or is restarting.
04 For native Windows, keep the PowerShell window open and scroll back to the first dependency, port, or permission error.
05 For macOS or Linux, copy the command, Python version, and first exception line before reinstalling packages.
06 Search logs for ChromaDB, MemoryVectorStore, DEGRADED, port, address already in use, permission denied, or module not found.
07 Do not paste full logs if they include API keys, tokens, cookies, email subjects, calendar events, file paths, or documents.
08 Redact secrets by replacing the middle with [REDACTED], not by deleting the entire context support needs.
09 If the same error repeats after three clean restarts, stop changing settings and prepare a support request.
10 Keep the failure reproducible: same command, same folder, same route, and one clear error.

18 Fix port and localhost problems

01 Connection refused means the browser cannot reach a running server at that address and port.
02 Address already in use means something else owns the port; stop that process or move Odysseus to another port.
03 On Docker, change APP_PORT in .env and recreate the container instead of editing generated container files.
04 On native uvicorn, change the --port value only after noting the old working or failing value.
05 Use 127.0.0.1 for same-machine access because it keeps the app private to the local computer.
06 Use localhost and 127.0.0.1 checks separately if one works and the other does not due to host resolution.
07 If Chrome caches a failed page, try a new tab with the exact URL rather than changing app settings blindly.
08 If a VPN, proxy, or security suite intercepts local traffic, temporarily test with it disabled only if you understand the risk.
09 If Docker publishes a port but the app inside uses another port, inspect compose logs and port mappings before guessing.
10 Write the final working URL in the handoff notes so the user opens the right page next time.

19 Match hardware to expectations

01 Separate Odysseus app requirements from local model requirements; the model is usually the heavy part.
02 A small CPU-only machine can still use external APIs or small local models while avoiding huge local downloads.
03 Check RAM, free disk, GPU vendor, GPU memory, and operating system before promising a local large-model experience.
04 Start with a small model even on strong hardware so network, model path, and endpoint setup are proven first.
05 If a model does not fit VRAM, lowering app settings will not magically make the model fast or stable.
06 On Apple Silicon, native local runtimes can use Metal paths that Docker Desktop cannot expose.
07 On Windows, advanced local GPU serving often means WSL2 or Linux routes, not the basic native app launcher.
08 Close games, video editors, duplicate model servers, and unused containers before benchmarking response speed.
09 Use one repeatable prompt when comparing speed so you do not blame random prompt complexity on setup.
10 If hardware is weak, recommend API provider setup instead of wasting support time on impossible local GPU promises.

20 Use files, tools, and agents safely

01 Do not connect personal files, email, calendar, shell tools, or browser automation until chat and model routing work.
02 Treat tool access as permissions, not features; each connector expands what the AI can see or affect.
03 Use a small test folder with harmless files before pointing Odysseus at real work documents.
04 Do not upload identity documents, legal files, customer data, private photos, or secrets during first setup.
05 If a tool can run shell commands, keep it disabled until you understand exactly what it may execute.
06 If an MCP or connector guide asks for tokens, create least-privilege tokens where possible and store them privately.
07 Test one connector at a time and write down the success condition before enabling the next connector.
08 If search or deep research returns weak results, verify the provider, network access, and rate limits before changing models.
09 Keep generated documents and memory stores backed up if they matter; do not assume reinstall keeps local data.
10 When troubleshooting, disable optional tools first so the base app and model route can be isolated.

21 Protect email and calendar connectors

01 Connect email or calendar only after the app, login, model, and basic chat are confirmed working.
02 Use a dedicated test account when possible instead of your main mailbox for first connector testing.
03 Read the requested scopes before approving OAuth; a calendar read scope is not the same as mailbox write access.
04 Do not send OAuth tokens, app passwords, recovery codes, cookies, or session screenshots to support.
05 Start with read-only behavior where the connector supports it; upgrade permissions only when a workflow needs it.
06 Test with a harmless calendar event or email thread before using real customer, financial, legal, or medical content.
07 If OAuth fails, capture the redirect URL domain, error code, and app setting name, but redact tokens.
08 If mail sync is slow, separate provider limits from Odysseus behavior by checking provider dashboards or rate-limit messages.
09 Disable or revoke connector access when selling the machine, handing off the install, or ending a test.
10 Document which account was connected so support does not confuse personal, business, and test identities.

22 Open from phone or another device

01 Do not search for an Odysseus APK as the first mobile route; phone access is normally browser access to your own instance.
02 Finish desktop login first before trying iPhone, Android, tablet, or another computer.
03 For same Wi-Fi access, bind intentionally to a LAN address and keep authentication enabled.
04 Use the computer's LAN IP only on trusted home or office networks; public cafe Wi-Fi is the wrong place.
05 For private remote access, prefer a VPN-style route such as Tailscale Serve over exposing a raw app port.
06 If using Tailscale, verify both devices are in the same tailnet before debugging Odysseus.
07 Use Tailscale Serve for tailnet-private sharing and understand that Funnel is public internet exposure.
08 If the phone opens the page but login fails, check cookies, HTTPS expectations, and whether the URL changed.
09 Do not leave APP_BIND=0.0.0.0 after testing if you no longer need LAN access.
10 Write down the phone-access route separately from desktop localhost because the URLs and risks are different.

23 Avoid unsafe public exposure

01 Never publish a raw Odysseus localhost port to the open internet just to make phone access convenient.
02 Use HTTPS, authentication, and an access layer before any domain or tunnel points at the app.
03 Cloudflare Access can protect a self-hosted app with policies, identity providers, sessions, and token checks.
04 Cloudflare Tunnel should be paired with Access before publishing a private app route to the internet.
05 Tailscale Serve is private to the tailnet; Tailscale Funnel exposes the service publicly and must be treated differently.
06 If you use Funnel for a demo, turn it off when the demo ends and confirm status from the CLI.
07 If you use Cloudflare Access, test denied access from a non-allowed account before trusting the setup.
08 Keep admin auth enabled inside Odysseus even when an external access layer is present.
09 Do not expose file, email, calendar, shell, or agent tools publicly during first setup.
10 If you cannot explain the public access path in one paragraph, keep the app local and ask for review before publishing.

24 Back up, update, and roll back

01 Before updating, save the working URL, start command, route, model endpoint, and any private configuration notes.
02 Back up .env and app data according to the README or project layout; do not back up venv or node_modules style artifacts.
03 For Git installs, update only from the official repository and read recent README changes before rerunning setup.
04 For ZIP installs, treat each ZIP as a snapshot and avoid merging old and new files by hand.
05 For Docker updates, rebuild and recreate containers after pulling changes so image and runtime match.
06 For native updates, rebuild the venv only when dependencies changed or dependency errors justify it.
07 Keep a copy of the last known working command before trying a new script, model server, or network exposure.
08 If update breaks startup, roll back the last changed thing first: port, provider, .env, model, dependency, or code.
09 Do not delete data folders to fix a dependency error unless the support plan explicitly says data reset is acceptable.
10 After repair, write a short handoff note: what failed, what fixed it, and what should not be changed next.

25 Improve performance after it works

01 Do not tune performance before the app can log in, connect a model, and answer a short prompt.
02 Benchmark one fixed prompt with one fixed model before changing context length, quantization, or provider.
03 Use smaller models for weak machines instead of forcing a large model that swaps memory and appears frozen.
04 Move model storage only when disk space is the bottleneck; it does not solve wrong endpoint or login failures.
05 Close duplicate Ollama, Docker, vLLM, SGLang, and browser-heavy sessions before measuring speed.
06 Separate first-token latency, tokens-per-second, and total answer quality so you know what problem you are solving.
07 If using an external API, check provider status, rate limits, and key billing before blaming local setup.
08 If document workflows are slow, test plain chat first, then one small document, then a realistic document batch.
09 If memory features degrade, check logs for vector store warnings before changing model providers.
10 Stop tuning when the user's actual task is acceptable; endless model swapping wastes paid support time.

26 Recover common broken states

01 If the browser cannot connect, check server process, port, and logs before reinstalling.
02 If Docker says the engine is unavailable, fix Docker Desktop or WSL2 before touching Odysseus.
03 If pip install fails, capture Python version, package name, and first compiler or dependency error.
04 If ChromaDB behaves oddly, check for incompatible chromadb-client conflicts before blaming memory features.
05 If admin password is missing, search the terminal and local logs before deleting configuration.
06 If Ollama model is missing, run ollama list and confirm the model name exactly matches settings.
07 If Docker cannot reach host Ollama, switch localhost to host.docker.internal as the route requires.
08 If phone access fails, verify desktop works, bind address is intentional, firewall allows it, and both devices share the route.
09 If email/calendar sync fails, revoke and reconnect only after checking scopes, redirect settings, and provider errors.
10 If everything is confused, return to the last known route and reproduce one failure from a clean terminal.

27 Prepare a safe support request

01 Send the payment email and package name so support can match the Stripe order.
02 Include the operating system exactly, such as Windows 10 Pro 64-bit, Windows 11, macOS Apple Silicon, Ubuntu, or WSL2.
03 State the install route: Docker, native Windows launcher, native macOS script, native Linux, or unsure.
04 Paste the exact command that failed, but remove secrets, tokens, private paths, and personal document names.
05 Paste the first real error line and up to 20 surrounding lines, not thousands of repeated retries.
06 Tell support whether http://localhost:7000, http://127.0.0.1:7000, or the macOS port opened.
07 Tell support whether Docker Desktop, Git, Python, and Ollama are installed and which version checks passed.
08 Tell support which model route you chose: local Ollama, external API, or no model yet.
09 List what you already tried so support does not repeat the same restart, reinstall, or port check.
10 Do not send passwords, API keys, cookies, private SSH keys, OAuth tokens, seed phrases, or full screenshots with sensitive data.

28 Know what the starter pack includes

01 The starter pack includes a detailed self-service SOP, email delivery, PDF backup, and basic support intake guidance.
02 It helps users choose a route, avoid fake downloads, complete first launch, and collect useful error details.
03 It does not include unlimited remote desktop time, custom infrastructure, custom code changes, or team deployment.
04 It does not guarantee local large-model performance on hardware that cannot run the chosen model.
05 It does not require the user to share secrets; any setup needing secrets should have the user type them privately.
06 A remote install package should define one machine, one app install, one model route, and one handoff note.
07 Public domain, Cloudflare Access, Tailscale, team accounts, and connector hardening are higher-risk support work.
08 GPU serving, vLLM, SGLang, CUDA, ROCm, and advanced Linux tuning need separate scoping before promising results.
09 If the customer only needs the free app, point them to the official GitHub README and avoid overselling services.
10 If the customer is blocked after the SOP, escalate with clean logs and a narrow next action instead of broad troubleshooting.

29 Run paid remote installation cleanly

01 Verify payment, package, email, and target domain before offering any remote session.
02 Confirm the user owns the computer and has permission to install developer tools, Docker, Python, and local AI software.
03 Tell the user they type all passwords and API keys themselves while you look away or pause screen share.
04 Start with a preflight checklist: OS, disk space, RAM, GPU, Git, Python, Docker, Ollama, browser, and network.
05 Keep the scope to one route unless the first route is impossible for a clear reason.
06 Narrate every command before running it so the user understands what changes on the machine.
07 Do not download files from unofficial mirrors during a paid session; use official project and vendor pages only.
08 At the end, show the user how to start, stop, update, and find logs without support present.
09 Send a handoff note with route, URL, model endpoint, start command, stop command, and safe support boundaries.
10 If the session uncovers a larger deployment need, stop and quote a separate package instead of silently expanding scope.

30 Final handoff and maintenance

01 Open the app once after a full stop and restart so the user knows the setup survives a reboot-style cycle.
02 Confirm the saved URL, login, model response, and one practical prompt before calling the install complete.
03 Show where logs live for the chosen route: Docker logs, PowerShell output, Terminal output, or Ollama logs.
04 Show how to update from the official repository and when not to update during urgent work.
05 Show how to disable LAN, Tailscale, Funnel, or Cloudflare exposure if it was enabled for testing.
06 Confirm no passwords, API keys, OAuth tokens, or payment details were collected by support.
07 Give the user a simple next-day checklist: start server, open URL, confirm model, run one prompt, stop cleanly.
08 Write down unresolved risks honestly, such as weak hardware, no GPU, unstable network, or public-access work not completed.
09 Keep the support channel tied to the payment email so future messages can be matched to the order.
10 Close with the official source reminder: future installs, updates, and repairs should begin from the GitHub README.

Use the official README for commands when it changes; use this PDF for the checklist discipline.

Windows click-by-click fallback checklist

01 Open Chrome on the Windows computer, type the official GitHub repository URL, and verify the owner before downloading anything.
02 Open File Explorer, create C:\OdysseusSetup, and keep every command and downloaded project file inside that folder.
03 Press the Windows key, type PowerShell, open a normal PowerShell window, and run cd C:\OdysseusSetup before cloning.
04 Run git --version; if it fails, install Git for Windows, close PowerShell, reopen it, and rerun the same version check.
05 Run py --list; if no Python 3.11+ appears, install Python, enable PATH if the installer offers it, then reopen PowerShell.
06 Clone the official repository once, run cd odysseus, and confirm launch-windows.ps1 is visible before running it.
07 Start the Windows launcher from inside the odysseus folder and leave the terminal open until a local URL or error appears.
08 If Windows blocks script execution, use the route recommended by Python and PowerShell docs instead of disabling security globally.
09 When the local page opens, copy the temporary admin password privately and change it inside the app after first login.
10 If setup fails, send support only the command, Windows version, install route, and first error line; never send secrets.

Use the official README for commands when it changes; use this PDF for the checklist discipline.

Docker recovery checklist

01 Open Docker Desktop and wait for the engine-ready state before running any docker compose command.
02 From the project root, run docker compose ps to see whether the app is running, restarting, exited, or missing.
03 Run docker compose logs --tail=120 odysseus and read from the first meaningful error, not the last retry line.
04 If port 7000 is taken, set APP_PORT=7001 in .env and recreate the container rather than editing app source files.
05 If Docker cannot reach host Ollama, use the documented Docker-to-host endpoint instead of plain localhost.
06 If image build fails, copy the failed package name and the first build error before running repeated rebuilds.
07 If the container starts but the browser fails, compare published ports, APP_BIND, APP_PORT, and the browser URL.
08 Use docker compose down for a clean stop before changing .env, switching model endpoints, or retrying a failed route.
09 Do not delete volumes or data folders unless the user accepts data loss and support explicitly recommends a reset.
10 After recovery, save the working compose command, URL, model endpoint, and stop command in the handoff notes.

Use the official README for commands when it changes; use this PDF for the checklist discipline.

Model provider worksheet

01 Write the chosen model route as local Ollama, external OpenAI-compatible API, OpenRouter, Groq, Anthropic, or undecided.
02 For local Ollama, record the model name, endpoint, storage location, and whether the first small model answered successfully.
03 For Docker plus host Ollama, record whether the endpoint uses host.docker.internal and whether the container can reach it.
04 For external APIs, record provider name, base URL, model name, billing limit, and whether streaming is enabled.
05 Do not record the secret API key itself; record only where the user stored it and when it was tested.
06 If chat fails, test the provider directly outside Odysseus to separate provider billing errors from app configuration.
07 If responses are slow, record hardware, RAM, GPU, VRAM, model size, and whether other model servers are running.
08 If quota fails, fix provider billing or rate limits before changing Docker, Python, or app ports.
09 If the user needs privacy, prefer local models; if the user needs reliability on weak hardware, prefer paid API routes.
10 After the first successful prompt, stop changing providers and write the exact working provider route into the handoff.

Use the official README for commands when it changes; use this PDF for the checklist discipline.

Safe support email template

01 Subject: Odysseus Setup Starter Pack support - payment email - operating system.
02 Line 1: I paid with this email address and I am using Windows, macOS, Linux, WSL2, or Docker.
03 Line 2: My install route is Docker, native Windows launcher, native macOS script, native Linux, or I am unsure.
04 Line 3: The exact command I ran was this command, with any private paths shortened or redacted.
05 Line 4: The first real error line is this line, plus at most 20 surrounding lines if needed.
06 Line 5: The browser URL I tried was localhost:7000, 127.0.0.1:7000, 127.0.0.1:7860, or another exact URL.
07 Line 6: Git, Python, Docker, and Ollama version checks passed or failed with these short results.
08 Line 7: The model route is not configured yet, local Ollama, or an external API provider.
09 Line 8: I already tried these steps, in this order, and stopped after the same failure repeated.
10 Final line: I confirm this email contains no password, API key, cookie, token, private key, seed phrase, or full secret log.

Use the official README for commands when it changes; use this PDF for the checklist discipline.

Remote install handoff record

01 Record the package, payment email, target site, and support scope before starting any remote install session.
02 Record the machine owner confirmation and whether the user has permission to install developer tools and local AI software.
03 Record the selected route, why it was selected, and which alternative routes were intentionally skipped.
04 Record exact install folder, start command, stop command, local URL, and admin login handoff instruction.
05 Record the model provider route without storing or transmitting the user's secret key.
06 Record whether phone access, LAN access, Tailscale, Funnel, Cloudflare Access, or public exposure was enabled or left disabled.
07 Record unresolved risks such as weak hardware, missing GPU runtime, unstable network, or public access not reviewed.
08 Record the first successful user-facing proof: login page, dashboard, model reply, document test, or connector test.
09 Record cleanup: temporary downloads removed, terminal stopped or left running intentionally, and no secrets collected by support.
10 Record next action: self-maintain, reply with safe logs, buy remote review, or scope a separate private deployment.

https://github.com/pewdiepie-archdaemon/odysseus | https://docs.docker.com/desktop/features/wsl/ | https://docs.docker.com/compose/install/ | https://docs.python.org/3.11/library/venv.html | https://docs.python.org/3.11/using/windows.html | https://docs.ollama.com/windows | https://tailscale.com/docs/reference/tailscale-cli/serve | https://tailscale.com/docs/features/tailscale-funnel | https://developers.cloudflare.com/cloudflare-one/access-controls/applications/http-apps/self-hosted-public-app/