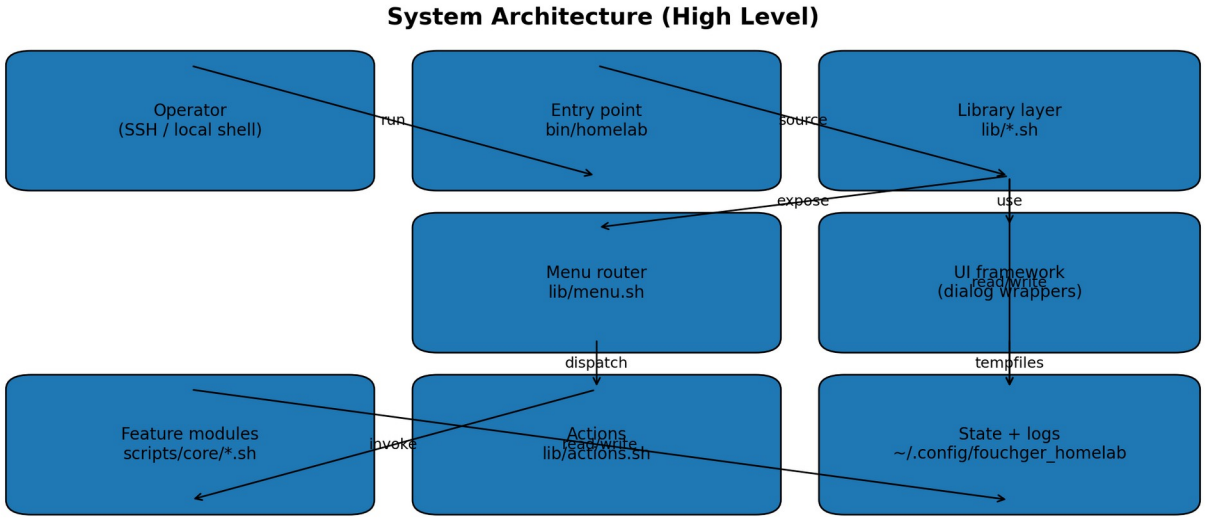
Fouchger Homelab

Developer Manual (v0.1) | 25 January 2026

This manual describes how the application is structured, how the codebase works, and how to extend it safely. It also includes a practical code review and recommended improvements.

1. Architectural overview

fouchger_homelab is a Bash-based, menu-driven CLI. The application follows a layered structure: entry point, library layer, menu routing, action orchestration, and optional feature modules. State and configuration are externalised to user-space files for portability.



2. Repository layout

Path	Purpose
bin/homelab	Entrypoint: loads libs and modules, optionally starts session capture, opens main menu.
lib/*.sh	Reusable libraries: paths, logging, core

	helpers, UI wrappers, feature flags, menu routing.
scripts/core/*.sh	Operational scripts and feature modules invoked from menus.
tests/*	Lightweight tests (currently repo root resolution).
install.sh / Makefile	Installer and developer convenience targets.

3. Startup sequence

On launch, bin/homelab anchors REPO_ROOT, loads libraries, loads feature modules, optionally starts Layer 2 capture, then enters main_menu().

1. Resolve and export REPO_ROOT.
2. Source lib/modules.sh then call homelab_load_lib (sources lib/paths.sh, logging, core, run, state, common, ui, features, actions, menu).
3. Call homelab_load_modules (best-effort sources scripts/core/app_manager.sh and other optional modules).
4. If FEATURE_SESSION_CAPTURE is enabled, attempt to start ptlog and show status.
5. Call main_menu (lib/menu.sh), which initialises UI and enters a selection loop.

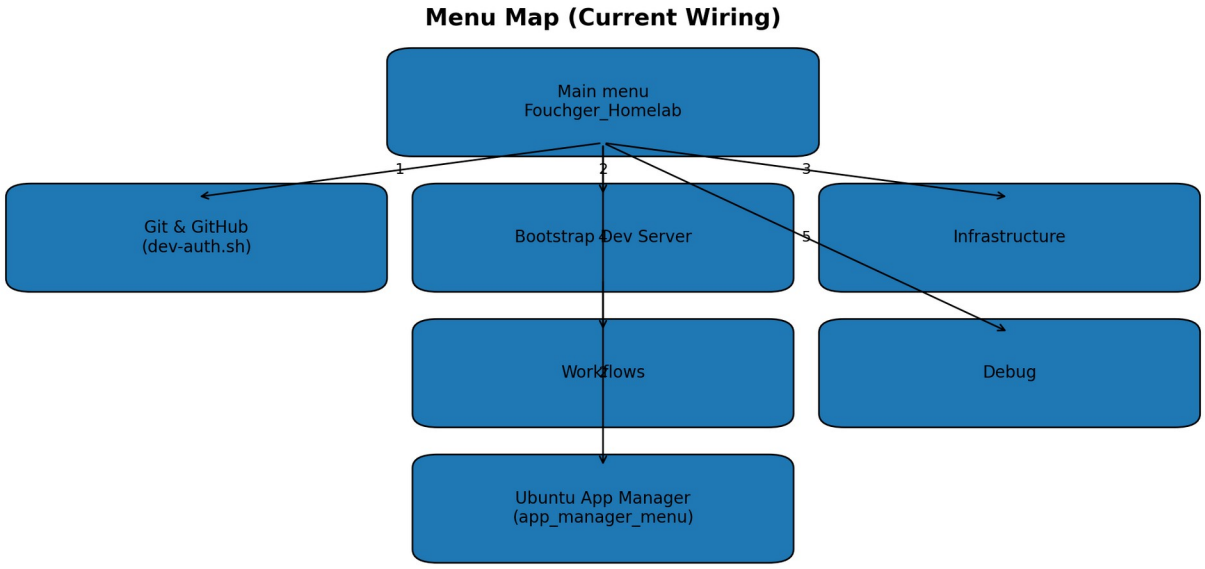
4. Design principles used in the codebase

- Strict mode (set -Eeuo pipefail) with controlled, explicit defaults.
- Best-effort optional modules: missing files do not break core runtime.
- UI separation: menu structure in lib/menu.sh; operational orchestration in lib/actions.sh; low-level operations in scripts/*.
- State externalisation: persist settings and selections under ~/.config/fouchger_homelab instead of inside the repo.
- Marker-based safety: uninstall operations should target only items installed by this tool.

5. UI framework and menu design

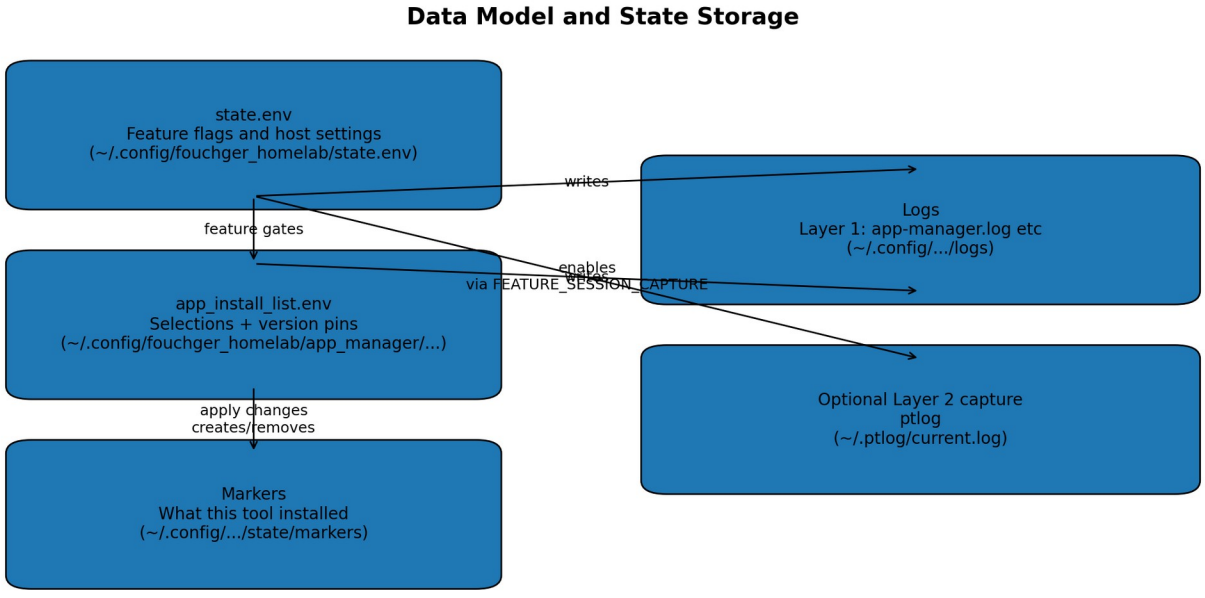
The UI uses wrapper functions in lib/ui.sh to avoid scattering direct dialog calls through the code. lib/menu.sh focuses on navigation and delegates work to action_* functions (or scripts) to keep responsibilities clean.

Current menu map:



6. State, configuration, and data model

The project uses dotenv-style env files for persistence. This keeps the runtime dependency footprint low and works well in LXC environments.



Core files:

- state.env: feature flags and host settings (lib/state.sh helpers).
- app_install_list.env: App Manager selections and version pins.
- markers: per-app files that indicate what was installed by this tool (used to make uninstall safe-by-default).
- Layer 1 logs: structured log files under ~/.config/fouchger_homelab/logs and app_manager/app-manager.log.
- Layer 2 capture: optional ptlog logs under ~/.ptlog.

7. App Manager internals

scripts/core/app_manager.sh is a self-contained module that exposes app_manager_menu. It maintains a catalogue of apps (APP_CATALOGUE) and a set of profiles that map to app keys. Selections are persisted to app_install_list.env as APP_<KEY>=0/1 along with version variables.

7.1 Strategies

Each catalogue item declares a strategy that determines how it is installed and how version pins are applied. Examples include apt, hashicorp_repo, github_cli_repo, binary, nvm, mongodb_repo, yq_binary, sops_binary, and docker_script.

7.2 Apply flow

6. Load env and selections.
7. Compute desired selections vs currently installed-by-tool markers.
8. Install selected items that are not installed.
9. Remove items that are no longer selected (but only when marker indicates installed-by-tool).
10. Write updated env and log outcomes.

8. API reference (functions and entry points)

This reference is generated from the current repository snapshot and lists Bash functions discovered in each file. Use it as a navigation aid rather than a formal interface contract.

bin/homelab

- maybe_start_session_capture

install.sh

- apt_install
- clone_or_update
- die

- ensure_deps
- have_cmd
- have_sudo
- info
- install_ptlog
- is_debian_like
- is_root
- log
- main
- need_cmd
- on_err
- post_clone_fixups
- require_sudo
- run_entry
- validate_config
- warn

lib/actions.sh

- action_open_app_manager
- action_open_dns_menu
- action_open_mikrotik_menu
- action_open_proxmox_templates
- action_run_questionnaires
- require_function
- source_if_exists

lib/common.sh

- _homelab_ensure_repo_marker
- _homelab_has_func
- homelab_die
- homelab_error
- homelab_info
- homelab_warn
- resolve_repo_root

lib/core.sh

- apt_install
- get_os_id
- get_os_like
- have_cmd
- is_debian_like

- is_root
- need_cmd
- require_sudo

lib/features.sh

- feature_enabled
- feature_key
- feature_require

lib/logging.sh

- _supports_colour
- error
- fg
- get_rgb
- info
- logging__layer1_emit
- logging__layer1_write_file
- logging_begin_capture
- logging_end_capture
- logging_rotate_file
- logging_set_files
- logging_set_layer1_file
- logging_strip_ansi_stream
- ok
- uc
- warn

lib/menu.sh

- app_manager_menu
- bootstrap_dev_server_menu
- debug_menu
- infrastructure_menu
- main_menu
- workflows_menu

lib/modules.sh

- homelab_load_lib
- homelab_load_modules
- source_if_exists

lib/paths.sh

- _find_repo_root_by_marker
- _homelab_ensure_repo_marker
- _homelab_paths_error
- _homelab_paths_warn
- ensure_dirs
- resolve_repo_root

lib/run.sh

- run_init
- run_on_error
- run_on_exit

lib/state.sh

- _state_key
- state_get
- state_init
- state_set

lib/ui.sh

- ui_checklist
- ui_confirm
- ui_ensure_dialog
- ui_exit
- ui_init
- ui_input
- ui_menu
- ui_msgbox
- ui_placeholder
- ui_run
- ui_textbox
- ui_tmpfile

scripts/core/app_manager.sh

- _app_test_show_dialog
- _catalogue_get_row_by_key
- _dpkg_is_installed
- all_app_keys
- apm_init_paths
- apply_changes
- apply_profile_add

- apply_profile_replace
- apt_pkg_has_candidate
- as_root
- audit_selected_apps
- backup_env_file
- choose_and_apply_profile_add
- choose_and_apply_profile_replace
- create_docker_remove_script
- default_for_version_var
- detect_os
- detect_pkg_mgr
- edit_version_pins
- ensure_apt_keyrings_dir
- ensure_cmd
- ensure_github_cli_repo
- ensure_grafana_repo
- ensure_hashicorp_repo
- ensure_mongodb_repo
- ensure_nodesource_repo
- ensure_pkg_mgr
- ensure_state_dirs
- filter_installable_apt_pkgs
- get_selection_value
- github_latest_tag
- install_by_strategy
- install_docker_via_get_docker
- install_helm_binary
- install_kubectl_binary
- install_python_target
- install_sops_binary
- install_yq_binary
- is_marked_installed
- is_pkg_installed
- is_python_version_target
- key_to_var
- known_version_vars
- load_env
- log
- log_line
- main
- mark_installed
- marker_get_field

- marker_get_packages_compat_csv
- marker_path
- need_cmd_quiet
- pkg_autoremove
- pkg_install_pkgs
- pkg_remove_pkgs
- pkg_update_once
- pkgs_csv_to_array
- profile_all_keys_unique
- profile_keys_for_name
- profile_version_lines_for_name

(Truncated: 89 total functions in this file.)

scripts/core/bootstrap.sh

- No Bash functions detected.

scripts/core/dev-auth.sh

- already_has_git_identity
- authenticate_gh
- configure_git_identity
- ensure_tool
- is_gh_authenticated
- is_tty
- main
- usage

scripts/core/make-executable.sh

- No Bash functions detected.

scripts/core/session_capture.sh

- flag_get
- flag_set
- ptlog_installed
- show_status

tests/test_repo_root.sh

- cleanup
- fail

9. Setup and configuration

9.1 Local development setup (Ubuntu)

11. Install prerequisites: git, make, dialog (and optionally gh).
12. Clone the repo and run make menu to verify the UI starts.
13. Run make executable (or scripts/core/make-executable.sh) after pulling changes on systems that strip executable bits.
14. Use scripts/core/bootstrap.sh for minimal bootstrap on fresh hosts.

9.2 Host configuration

Feature flags live in state.env. Example enablement commands:

state_set FEATURE_SESSION_CAPTURE 1
state_set FEATURE_PROXMOX 1
state_set FEATURE_MIKROTIK 1
state_set FEATURE_DNS 1

10. Tutorials and guides

10.1 Add a new main menu item

15. Implement a new script or action function (prefer scripts/<area>/... for larger features).
16. Expose an action_* wrapper in lib/actions.sh if it improves decoupling.
17. Add a new option in lib/menu.sh main_menu() using ui_menu and dispatch to your action.
18. If the feature is host-dependent, gate it behind feature_require in lib/features.sh.
19. Update documentation and add a smoke test if possible.

10.2 Add a new App Manager application

20. Add a new APP|... line to APP_CATALOGUE, giving it a unique key.
21. Choose an installation strategy (or add a new one with clear guardrails).
22. If version pinning is needed, add a version var and include it in known_version_vars().
23. Update profiles if this app belongs in one of the baseline bundles.
24. Add or update audit logic so 'check which apps are installed' remains trustworthy.

10.3 Add a new feature flag

25. Pick a consistent flag name (FEATURE_<NAME>).
26. Use state_set/state_get via lib/features.sh helpers where possible.
27. Gate menu entry points with feature_require to provide a friendly enablement message.
28. Document the flag and add a default behaviour when missing.

11. Code review findings and recommendations

11.1 What is working well

- Consistent strict-mode usage across most files improves runtime safety.
- Clear separation of menu routing (lib/menu.sh) from operations (lib/actions.sh and scripts).
- Repo root resolution and state storage design support installs without a .git directory, which is pragmatic for copied deployments.
- Two-layer logging is a strong operational pattern for troubleshooting in homelabs.
- App Manager catalogue and profile approach is scalable for repeatable builds.

11.2 Key risks and improvement opportunities

- Third-party script execution: bootstrap_dev_server_menu runs an external curl|bash installer. Consider checksums, pinning to a specific commit/tag, and logging the exact source URL and version.
- Feature loading clarity: bin/homelab comments out direct sourcing of state/features; rely on homelab_load_lib. This is fine, but keep the entrypoint tidy to avoid confusion.
- Dependency detection: dialog availability is assumed. Consider a preflight that checks for dialog and offers to install it (or falls back to a non-UI mode).
- Marker model: markers are currently stored under ~/.config/.../state/markers. The App Manager comments suggest /usr/local/share markers. Decide on one and document it to avoid surprises.
- Idempotency and rollback: package installs are idempotent, but repo additions (HashiCorp/Mongo/Grafana) benefit from explicit cleanup paths and backup of repo list files.
- Testing: only minimal tests exist. A lightweight CI check could run bash -n, verify functions load, and validate env file writes in a temp HOME.

11.3 Suggested next steps (pragmatic roadmap)

- Add a preflight command: detect OS, dialog, sudo, and connectivity, then summarise readiness.
- Introduce a 'dry run' mode for App Manager apply to show the planned changes before executing.
- Add a simple changelog and versioning so user manuals align to a release tag.
- Provide a 'non-interactive' mode for App Manager driven entirely from env files for automation (Ansible friendly).
- Add a minimal GitHub Actions workflow to run bash syntax checks and basic unit tests.

12. Code commenting and file header standard

To keep the codebase consistent, use a standard header block at the top of each script and capture: filename, purpose, usage, key assumptions, and maintainer. Keep function-level comments focused on why and constraints rather than restating what the code does.

Recommended header template:

```
#!/usr/bin/env bash
# ------------------------------------------------------------------------------
# Filename: <path>
# Description: <one line>
# Usage: <how to run>
# Notes:
#   - <key constraints and guardrails>
# Maintainer: <name/team>
# ------------------------------------------------------------------------------
```

Document generated from repository snapshot: fouchger_homelab-main (25 January 2026).