Plan for Open-Sourcing the AllTuner Scaffolding

Executive Summary

- The current scaffolding is tightly coupled to the private AllTuner platform through defaults, infrastructure endpoints, and auxiliary tooling. Any public release must first extract a product-agnostic core and then offer the AllTuner profile as an optional plugin or reference implementation.
- Key themes for the transition: (1) parameterise infrastructure, (2) split vendor-specific helpers into optional packages, (3) ship a minimal public default experience, and (4) publish supporting tooling (`prototuner`, `startuner`) in a consumable way.
- Recommended roadmap: stabilise internal APIs, introduce a modular configuration system, release the generic core, and finally publish the AllTuner opinionated layer once legal/security checks are complete.

Current AllTuner-Specific Coupling

Template configuration & defaults

- `copier.yml:71` – default GitHub remote points to the `alltuner` org; redis and Mongo defaults reference private Tailscale hosts.
- `template/pyproject.toml.j2:11` & `pyproject.toml:8` – base dependency is `prototuner`, a private bundle with AllTuner-focused versions; public release requires either publishing `prototuner` or expanding dependencies inline.
- `template/src/{{ project_slug }}/core/config.py:44` – baked-in company identity (`All Tuner Labs`, `corp@alltuner.com`).

Docker & deploy automation

- `template/compose.prod.yml.j2:4` hard-wires the external network `alltuner-network` plus registry `alltuner-docker-registry.long-python.ts.net` and Cloudflare/Umami labels.
- `template/start.sh.j2` assumes Watchtower and Cloudflare R2-backed storage when the job queue flag is enabled.
- The generated just recipes (`template/justfile`) depend on Docker Buildx bake pushes to the private registry and optional SSH deploy host flows.

Platform services baked into the stack

- `template/src/{{ project_slug }}/services/core/email.py` is SES-only; credentials flow expects AWS IAM access.
- `template/src/{{ project_slug }}/services/core/blob.py` mandates Cloudflare R2 connection details; raises if not configured.
- `template/src/{{ project_slug }}/frontend/default_routes/debug.py` leaks internal model names (Stations, Rundowns) hinting at domain-specific

debugging utilities.
- `template/templates/frontend/defaults/...` assume Umami analytics (labels in `compose.prod.yml.j2`) and Cloudflare DNS automation.

External Repositories & Their Roles

Prototuner (`prototuner` repo)

- Currently the single source of truth for dependency versions (FastAPI, Beanie, Streaq, Authlib, etc.).
- Dev extras include Cloudflare SDK, Babel, Ruff, typed stubs. Needs a permissive licence and release schedule if it remains a dependency; otherwise fold the dependency list into the public template.

AllTuner Shared Infra (`alltuner-shared-infra`)

- Docker Compose environment provisioning Mongo, Redis, private registry, Umami, and Grafana through Tailscale sidecars. Public users will not have this network; provide optional "bring your own infra" docs and reference Terraform/Compose without the internal hostnames.

StarTuner (`startuner` repo)

- CLI wrapper around the scaffolding with Cloudflare DNS automation and Umami provisioning (`src/startuner/cli/__init__.py`, `cloudflare.py`, `umami.py`).
- Defaults assume GitHub org `alltuner`, destination host `papaya.alltuner.com`, and analytics endpoint `https://analytics.alltuner.com/api` (`src/startuner/defaults.py`).
- For open-source distribution, either: (a) publish a trimmed CLI without proprietary integrations, or (b) extract AllTuner-specific behaviour into optional subcommands guarded behind environment detection.

Recommended Refactor Strategy

1. **Define a Core Profile**
   - Extract AllTuner-specific defaults into a dedicated profile (e.g. `profiles/alltuner/*.j2`) while the base template uses neutral settings (`localhost` endpoints, generic company metadata, optional features off by default).
   - Introduce a copier question to select a profile (`all`, `minimal`, `alltuner`) and load profile-specific defaults.
2. **Parameterise Infrastructure & Third-Party Services**
   - Replace hard-coded network and registry names with copier variables (`docker_registry`, `docker_network`, `enable_watchtower`, `enable_caddy`) plus guard rails to skip sections when unset.

- Provide alternative storage/email providers by implementing provider interfaces (`EmailService`, `BlobStorageService`). Ship SES/R2 implementations in an `extras/alltuner` module, while default core uses local file backend or leave stubs with documentation.

3. **Rework Dependency Packaging**
   - Decide whether to release `prototuner` publicly (with versioned tags, licence, and docs). If not, copy dependency pins into the template and allow users to opt into an `alltuner` dependency group.
   - For the CLI, break the startuner package into `scaffolding-cli` (agnostic) and `startuner-alltuner` extension that wires Cloudflare/Umami automation.
4. **Documentation & Developer Experience**
   - Expand `README.md` with a full quick-start (requirements, copier invocation, local dev, optional Docker). Document feature flags and how to supply provider credentials via `.env`.
   - Provide migration notes for existing internal users to adopt the modular profile structure.
5. **Security & Compliance Sweep**
   - Remove or anonymise private hostnames, email addresses, and secret tokens before publishing history (consider history rewrite or repo squashing).
   - Finalise an SPDX-compliant licence (likely MIT or Apache-2.0) covering scaffolding, templates, and helper scripts.

Phased Roadmap

- **Phase 0 – Inventory & Issue Tracking**: File GitHub issues for each workstream (dependencies, infra abstraction, documentation, CLI split, licensing). Establish acceptance tests for generated projects to avoid regressions.
- **Phase 1 – Core Template Hardening**: Implement profile-aware copier config, neutral defaults, and provider interfaces. Ensure generated project works without AllTuner services (local Mongo/Redis optional, no analytics by default).
- **Phase 2 – Publish Supporting Packages**: Tag and release `prototuner` (or replace it), publish a pared-down CLI (`scaffolding-cli`) with installation instructions (Python package + `uv tool install`).
- **Phase 3 – Optional AllTuner Extension**: Add documentation describing how to enable the AllTuner profile, including references to private infra, ensuring secrets stay out of the public repo (deliver as a separate private package if necessary).
- **Phase 4 – Launch & Governance**: Write contribution guidelines, set up CI (lint, copier test copy, docker build), configure Dependabot/Renovate for dependencies, and announce the project with changelog + release notes.

Additional Considerations

- **Testing Matrix**: Add automated copier smoke tests (generate project via GitHub Actions matrix for Python versions 3.10–3.13, with and without optional features).
- **Frontend Assets**: Consider publishing the default HTMX/Tailwind theme as a separate package so other users can opt-out cleanly.
- **Background Jobs**: Document expectations around Redis/Streaq; optionally turn off queue scaffolding unless explicitly enabled.
- **Analytics & Observability**: Provide guidance on third-party alternatives (PostHog, Plausible) or leave placeholders in the template metadata.
- **Community Support**: Set up Discussions or Slack/Discord to gather feedback once open-sourced. Prepare a code of conduct and security policy.