Skynet — Task specs (S1, executable from the text alone) — S1-rev

---

For: Agon, Sead, Sven (CTO), Vince — lanes: A = Agon, B = Vince, C = Sead

Standard per card: Context · Read first (paths) · Deliverables (files, endpoints, tables, routes by name) · Interfaces (consumes / exposes) · Steps · Acceptance (a command or artifact someone else can verify) · Out of scope · Waits on / Unblocks.

Repo conventions used below: monorepo `Rescale-Media/skynet`; `apps/api` = the static-ads FastAPI backend (today `static-ads-automation/app/backend` + `app/pipeline`), `apps/web` = its Vite/React frontend (today `app/frontend`), `packages/contracts`, `services/scraper|transcriber|agent-runtime|workos`, `infra/`, `db/migrations/unified/<schema>/NNN_*.sql`, `docs/`. Until A1a lands, paths below map 1:1 to the source repos named in each card. Tests: `pytest apps/api/tests`, `pnpm --filter <pkg> test`, `pnpm vitest` in workos. Sizes: S ≤ ½ d · M ≈ 1 d · L ≈ 2 d.

Lane C — Sead (Data spine & Bridge)

WAVE	KEY	CARD	SIZE	WAITS ON	UNBLOCKS
W0	**S0a**	Scrubbed-import prep for workos	S	—	A1a (A)
W1	**S1a**	Core auth + workspaces + contract	M	—	A3 (A), A4 (A), A6 (A), M1 (V), M2 (V), S13 (S), S1b (S), S3 (S), S4 (S), S4b (S), S5a (S), S8 (S), S8a (S), V1 (V), V13 (V), V2a (V), V4 (V), X1 (A)
W1	**S1b**	Invite/magic link/reset + user import	M	S1a (S)	M3 (A)
W2	**S3**	Creative-code service	M	S1a (S), A1a (A)	S13 (S), S5a (S), S7 (S), S8 (S), S8a (S), V13 (V), V7 (V)
W2	**S4**	Lineage + outbox (minimal)	S	S1a (S)	S13 (S), S7 (S), V10 (V), V13 (V), V7 (V)
W2	**S4b**	Bindings/spend-guard contract artifact	S	S1a (S)	V13 (V), V4 (V)
W3	**S5a**	Ecomprofits read layer	M	S1a (S), S3 (S)	S7 (S)
W3	**S8a**	Workos bridge endpoints: ad-name assembly + attach export package to the Launch card	M	S1a (S), S3 (S)	S10 (S)
W4	**S10**	Creative-code loop E2E with one real ad	M	V7 (V), S8a (S), S7 (S)	S11 (S)
W4	**S7**	Readback job v0	M	S5a (S), S3 (S), S4 (S)	S10 (S), S13 (S), V10 (V)
W5	**M5**	Retention / archive plan	S	X5 (A)	—
W5	**S12**	Migration dry run (production PG) + rollback scripts	M	V3 (V), A6 (A)	S11 (S)
W5	**S13**	Cross-tenant isolation tests	S	S1a (S), S3 (S), S4 (S), S7 (S), V4 (V), V10 (V)	—
S4	**S0b**	Pre-pilot discovery (per-brand launch process)	M	—	S6 (S)

WAVE	KEY	CARD	SIZE	WAITS ON	UNBLOCKS
S4	**S11**	Pilot SOP + walkthrough → G3	S	S6 (S), S8 (S), V9 (V), S10 (S), X5 (A), S12 (S)	—
S4	**S6**	Pre-pilot discovery write-up (G2)	S	S0b (S)	S11 (S), S8 (S)
S4	**S8**	S8b · Launch-board API + ClickUp mirror (S4)	L	S1a (S), S3 (S), S6 (S), A1a (A)	S11 (S), V9 (V)

S0a · Scrubbed-import prep for workos · S · `chore/s/s0a-scrub-workos` (with Sven)

Context. The rescale-workos repo has live credentials committed on purpose (co-founder handoff). The monorepo import (A1a) must not carry one of them. Decision already taken: no rotation now; credentials are discontinued when workos leaves Railway (S17, S4).

Read first. `rescale-workos/CLAUDE.md` lines 40–82 (`## Access and credentials` : ClickUp, AI/comms, Google, Postgres, Railway); `rescale-workos/.gitignore` lines 1–11 (says credentials are committed; `secrets/` is NOT ignored); `rescale-workos/secrets/` (≈110 tracked files — `SESSION-PRIMING.md` , `google-service-account.json` , `field-create.curl` , `automation-group.curl` , `apply-migration-00{2,3,4,6}.mjs` , `ecomprofits-role-test.mjs` , `reseed-globals-2026-07-09.ts` , `p*/w*-*.mjs` probes); `rescale-workos/service/.env.example` (the env names, no values).

Deliverables. (1) `docs/security/workos-credential-inventory.md` in the *old* repo (private): one row per credential — name, where it lives today (CLAUDE.md line / secrets file / Railway variable), owner, discontinuation date S17. No values in the doc. (2) Under `infra/secret-scan/` , create both `gitleaks.toml` and `patterns.txt` . `gitleaks.toml` is the single config used by both the manual scan and the pre-commit hook; it must include the stock gitleaks rules plus the custom patterns from `patterns.txt` (one custom regex per known value: first 8 characters + length, never the value). (3) The scrubbed working tree: `rescale-workos` checkout with `secrets/` removed, `CLAUDE.md` lines 40–82 replaced by one paragraph "Credentials live in Railway variables and the 1Password vault *Rescale workos*; the old repo stays private as the archive until S17", `.gitignore` gaining `secrets/` , `*.pem` , `*.key` kept. Hand the tree's path to Agon. (4) Pre-commit hook config (`.pre-commit-config.yaml` with gitleaks using the pattern file) ready for A1b.

Interfaces. Exposes the scrubbed tree + pattern file to A1a/A1b.

Steps. Inventory with Sven (30 min) → write patterns → `gitleaks detect --source <scrubbed-tree> --config infra/secret-scan/gitleaks.toml` → fix until zero hits → grep the tree for each first-8-chars → hand over.

Acceptance. Paste in the TechDesk task: (a) `gitleaks detect` output on the scrubbed tree = 0 findings; (b) a deliberate test commit containing one known value is rejected by the pre-commit hook (terminal output); (c) inventory doc link.

Out of scope. Rotating anything; Railway changes; the monorepo import itself (A1a).

S1a · Core auth + workspaces + contract · M · `feat/s/s1a-core-auth`

Context. One login for every surface. Core issues RS256 tokens; every other service only verifies. This is the first cross-lane blocker: V1, V2a, V6b, A3, A5-lite all consume it.

Read first. Lock doc §4.1 (the contract — claims, cookies, refresh rotation, `tv` logout propagation, service tokens, legacy mapping); static-ads `app/backend/auth.py` (today's `create_token`, `get_current_user` L61–67 reading Bearer or `?token=`, `token_version` check L74–88); `app/backend/main.py` L75 (`auth.router` at `/api/auth`) and L78 (`auth_dep`); `app/backend/db.py` (`get_pool`, `get_cursor`); `schema.sql` `users` table.

Deliverables.

1. Migration `db/migrations/unified/001_core.sql`: `core.workspaces(id uuid pk, slug unique, name, settings jsonb, created_at)`, `core.users(id uuid pk, email citext unique, name, password_hash, token_version int default 0, is_active bool, created_at)`, `core.permissions(key text pk, module, action, description)`, `core.roles(id uuid pk, workspace_id uuid null, key, name, is_system bool default false; unique key where workspace_id is null; unique (workspace_id, key))`, `core.role_permissions(role_id, permission_key, pk both)`, `core.workspace_members(workspace_id, user_id, pk both)`, `core.workspace_member_roles(workspace_id, user_id, role_id, pk all three)` — roles are data, a member may hold several, effective permissions = **union of allows** (no deny rules), `core.refresh_tokens(id uuid pk, user_id, family_id uuid, token_hash, expires_at, revoked_at, replaced_by uuid)`, `core.feature_flags(workspace_id, key, enabled bool, pk(workspace_id,key))`, `core.schema_migrations(version text pk, checksum text, applied_at)`. Seed: workspace `rescale`, admin user from `ADMIN_EMAIL` / `ADMIN_PASSWORD`; the six system roles `admin, strategist, media_buyer, editor, researcher, viewer` and the S1 permission catalogue: `admin.workspace.manage, admin.members.manage, admin.settings.manage, admin.audit.read, intelligence.read, intelligence.write, intelligence.teardown.run, intelligence.research.promote, studio.read, studio.write, studio.approve, production.read, production.write, production.export, workspace.read, workspace.launch.write, performance.read` with the role → permission mapping in `packages/contracts/auth/README.md`.
2. Module `apps/api/core/auth/`: `keys.py` (load `AUTH_JWT_PRIVATE_KEY_PEM` / `PUBLIC`, `kid` = sha256 prefix), `tokens.py` (`issue_access(user, ws) → jwt`, `issue_refresh`, `rotate_refresh`, `revoke_family`, `issue_service(name, ws)`), `verify.py` (`verify_access_token(token) → Claims`, checks `kid`, `exp`, `typ`, `iss`, and `tv` against a 60-s cached lookup), `deps.py` (`current_user`, `require_permissions(*keys)` — every route gate checks permissions, never role names; `require_roles(*roles)` exists only as a thin compatibility shim during the port), `router.py`.
3. Routes under `/api/auth`: `POST /login {email,password}` → sets `rs_access` (12 h) + `rs_refresh` (30 d, path `/api/auth`) cookies, HttpOnly/Secure/SameSite=Lax, `Domain` = exact

host from `PUBLIC_HOST`; `POST /refresh` (rotation; reuse of a revoked token revokes the family); `POST /logout` (revokes family, bumps `token_version`, clears cookies); `GET /me` → `{id, email, name, ws, ws_slug, roles, permissions, feature_flags?}` for the current session (`roles` = the member's role keys; `permissions` = the effective union resolved server-side with a short cache — the canonical source for UI gating; the JWT carries roles only, never permissions) — the canonical shape V2a/V6 use for the workspace switcher and route gating; `GET /jwks`; `GET /session-token` (returns the current access JWT as JSON for the static-ads bridge, same-origin only); `GET /tv/:sub` (internal, service token); `POST /service-token` (admin only).

4. `packages/contracts/auth/`: `access-claims.schema.json`, `service-claims.schema.json`, `samples/` (valid, expired, wrong-kid, revoked-tv — signed with a test keypair committed under `samples/test-keys/`), `verify.py`, `verify.ts` (jose), `verify.node.cjs`, and `README.md` (claim table + cookie table + legacy role mapping).

Interfaces. Exposes the cookie names, claims, JWKS, `/session-token`, `/tv/:sub`. Consumed by V1, V6b, A5-lite, A3, X1.

Steps. Migration → keys/tokens → routes → contracts package + samples → tests → README.

Acceptance. `pytest apps/api/tests/core/test_auth.py` green (login sets both cookies; refresh rotates; reuse revokes family; logout bumps `tv`; `/me` 401 after logout); `python packages/contracts/auth/verify.py packages/contracts/auth/samples/*.jwt` and `node packages/contracts/auth/verify.node.cjs packages/contracts/auth/samples/*.jwt` print the same 4 verdicts (valid / expired / wrong-kid / revoked-tv). Post both outputs.

Out of scope. Invite/magic link/reset (S1b); users UI (V11); RLS.

S1b · Invite / magic link / reset + user import · M · `feat/s/s1b-users`

Context. Existing users of static-ads, the scraper admin, and the console must sign in at the S1 exit gate without a new password ceremony run by hand.

Read first. S1a routes; static-ads `schema.sql` `users` (email, display_name, is_admin); scraper `meta_ads.admin_users` (email, role admin|user); console `users` table (`RescaleOS/console/src/db/client.ts`, roles in `src/lib/roles.ts`); lock doc §4.1 legacy role mapping.

Deliverables. (1) `core.auth_tokens(id, user_id, kind ∈ invite|magic|reset, token_hash, expires_at, used_at)`; routes `POST /api/auth/invite` (admin), `POST /api/auth/magic-link`, `POST /api/auth/reset/request`, `POST /api/auth/reset/confirm`, `GET /api/auth/accept?token=` (validates the token and returns `{email, kind, expires_at}` only), `POST /api/auth/accept {token, password}` (sets the password for invite/reset, marks the token used, signs in, sets both cookies). (2) `apps/api/core/scripts/import_users.py`: reads the three user tables (connection strings via env), dedupes by lower(email), creates `core.users` (inactive until accepted) + `workspace_members` with roles: static-ads `is_admin` → `admin`; if `INTELLIGENCE_OWNER_EMAIL` is set and the email is in that allowlist (static-ads derives `can_see_intelligence` from env, `auth.py` L33 — it is not stored), also add `researcher`; scraper

`admin` → `admin`, `user` → `researcher`; console `editor` → `editor`, `creative_strategist` → `strategist` (`RescaleOS/console/src/lib/roles.ts` L4); everyone → `viewer`. Writes `docs/ops/user-import-<date>.csv` (email, sources, roles, invite sent y/n). (3) Emails go through M3's mailer; until M3 lands, the script prints the accept URLs.

Interfaces. Consumes S1a; M3 for sending.

Steps. Table + routes → import script against dumps → dry-run report → tests.

Acceptance. `pytest apps/api/tests/core/test_users.py` green (invite → accept → login; expired token 410; reset flow); `python apps/api/core/scripts/import_users.py --dry-run` on the three real dumps prints N users, 0 duplicate emails, and the CSV; post the CSV summary (counts only).

Out of scope. Sending emails (M3); role editing UI (V11).

S3 · Creative-code service · M · contract-wave schema first · `feat/s/s3-creative-codes`

Context. The creative code is the wire that closes the loop: Production stamps it, the ad name carries it, ecomprofits parses it back. There is exactly one generator today, in workos. Skynet must produce byte-identical codes and must draw `seq` from the same counter, or codes collide.

Read first. Lock doc §4.2; workos `service/src/modules/naming/creative-code.ts` (`buildCreativeCode(parts)` L84, `variantLetter`, `kindFromFormat`, `isTeamMade`, `teamMadeFromCode`), `sequence.ts` (`nextSequence(db, productNumber, codeType)` L18 → `select rescale_service.next_naming_seq($1,$2)`), `migrations/001_sequence_counter.sql`; workos `assemble.ts` (`assembleAdName`, `AD_REQUIRED` L93–109) and `__fixtures__/naming-conventions.ts` (`META_AD_V3`); ecomprofits `packages/features/naming-conventions/src/lib/parser.ts` (`parseNameWithRules(sourceName, rules)` L18) and the live rule set for the Rescale Meta account (export it with `select rules from naming_conventions where account_id='48810a93-baae-4bc3-b562-5cf8814f0edf' and integration_type='meta' and entity_level='ad' and is_active;` and save it as `packages/contracts/creative-code/ecomprofits-meta-rules.json`).

Deliverables. (1) Migration `db/migrations/unified/core/002_creative_codes.sql`: `core.creative_codes(code text, workspace_id uuid, product_number text, kind text check (kind in ('RC','TM')), team_made bool, seq int, variant text, asset_id uuid null, asset_key text null, asset_type text, session_id uuid null, issued_at timestamptz, issued_by uuid, naming_version text default 'v3', unique (workspace_id, code))`. (2) `apps/api/core/creative_codes/generator.py`: a line-for-line port of `buildCreativeCode` + `variantLetter` + `kindFromFormat` + `isTeamMade` (same errors). (3) `sequence.py`: `next_seq(product_number, kind)` executing `select rescale_service.next_naming_seq(%s,%s)` on the ecomprofits connection with the `rescale_service` role (env `ECOMPROFITS_WORKOS_URL`), inside the same transaction as the insert — no local counter, ever. (4) Routes: `POST /api/creative-codes {product_number, kind, team_made, variant?='A', asset_type, session_id?}` → `{code, seq}`; `GET /api/creative-codes/:code`; `POST /api/creative-codes/:code/iterate` → next variant letter, same seq. (5) `packages/contracts/creative-code/`: `schema.json`,

`normalize.py` / `normalize.ts` (`normalize_creative_code`: upper-case → `RCi` / `TMi` casing restored, `\s*(hook|_HK)\s*(\d)` → `hook N`, trim), fixtures `live-samples.json` (≥30 real `nc_creative_code` values from `meta_ads_dashboard_view`, including `RCI31-A_HK 1`), and `cross-check.json` (1,000 random `CreativeCodeParts` + the TS output). Add the helper script `service/scripts/dump-creative-codes.ts` in the workos repo **in this card** (it does not exist yet) to generate that fixture from `service/src/modules/naming/creative-code.ts`; after A1a the same file lives at `services/workos/scripts/dump-creative-codes.ts`.

Interfaces. Exposes the routes + the normaliser. Consumed by V7 (stamp), S7 (readback key), S8 (launch task field).

Steps. Contract wave: schema + fixtures merged first. Then generator port + cross-check test → sequence via ecomprofits → routes → parser round-trip test.

Acceptance. `pytest apps/api/tests/core/test_creative_codes.py` green: (a) 1,000/1,000 cross-check parts produce the TS string; (b) 20 issued codes placed into a v3 ad name (fixture `META_AD_V3` layout) parse with the exported ecomprofits rules and `creative_code` round-trips; (c) two concurrent issues for the same product get distinct `seq` (test against a scratch `rescale_service` schema); (d) normaliser maps every live sample to canonical form. Post the test output.

Out of scope. Ad-name assembly (workos endpoint, S8); RescaleOS ad names (A12, S3); changing the grammar.

S4 · Lineage + outbox (minimal) · S · `feat/s/s4-lineage-outbox`

Context. The S1 exit gate must show the chain session → concept → asset → export → code → performance for one real ad. Only Production (V7) and readback (S7) write to it in S1.

Read first. Lock doc §4 (contract 3); technical plan §2.4.3.

Deliverables. Migration `003_lineage.sql`: `core.lineage_nodes(id uuid pk, workspace_id, type text check (type in ('session','concept','asset','export','creative_code','launch','performance','winner')), ref_id text, attrs jsonb, created_at)`, `core.lineage_edges(from_id, to_id, relation text, pk(from_id,to_id))`, `core.events_outbox(id bigserial, workspace_id, type, payload jsonb, created_at, processed_at null)`. Module `apps/api/core/lineage/`: `POST /api/lineage/nodes` (batch: nodes + edges, idempotent on `(workspace_id,type,ref_id)`), `GET /api/lineage/chain?type=creative_code&ref=<code>` → ordered chain (both directions, depth ≤ 8), poller skeleton `outbox.py` (marks processed, no consumers yet). `packages/contracts/lineage/schema.json` + fixture chain.

Interfaces. Consumed by V7 (writer), S7 (writer), V10 (reader).

Acceptance. `pytest apps/api/tests/core/test_lineage.py`: insert the fixture chain → `GET /chain?type=creative_code&ref=TMi2-A` returns session→concept→asset→export→creative_code→performance in order; re-posting the same batch creates no duplicates.

Out of scope. Scraper/studio/workos producers, inbox, coverage store (S14/S15, S3).

S4b · Bindings / spend-guard contract artifact · S · `feat/s/s4b-bindings-contract`

Context. Vince builds V4 (tables + evaluation) and every sidecar will post usage. The contract must exist before the code so consumers build against fixtures.

Read first. Lock doc §4 contract 4; technical plan §2.4.4; V4 spec below (tables).

Deliverables. `packages/contracts/bindings/`: `binding.schema.json` (`GET /api/bindings/:workflow` response: `{workflow_key, provider:{kind, base_url}, model:{slug, modality}, config:{…}, paused:bool, guard:{window_hours, warn_usd, pause_usd, spent_usd}}`), `usage-event.schema.json` (`POST /api/usage`: `{workspace_id, service, workflow_key, model, tokens_in, tokens_out, cost_usd, request_id}`), `spend-guard-states.md` (`ok → warn → paused → resumed`, who can resume), fixtures (one per state), consumer tests: Python (`apps/api`), Node (`services/scraper`, `services/agent-runtime` — a 10-line client each that validates against the schema).

Acceptance. `pnpm --filter contracts test` green; the three consumer tests import the fixtures and pass.

Out of scope. The implementation (V4), the admin UI (V5).

S5a · Ecomprofits read layer · M · `feat/s/s5a-ecomprofits-read`

Context. Performance comes back from ecomprofits by creative code. If the columns or formulas are wrong, the "closed loop" is fake. All facts below are verified against the ecomprofits schema.

Read first. Lock doc §4.3 (contract); ecomprofits `apps/web/supabase/schemas/27-meta-ads.sql` (`meta_ad_insights` L383–484, `meta_intraday_ad_insights` L620, `combined_meta_ad_insights` L919, `meta_ads.parsed_naming` L211–244), `40-dashboard-views.sql` (readonly grant loop L304–312, role list at L309; klaviyo loop L1514–1522, L1519), `39-ads-performance-unified.sql`; `packages/features/meta-ads-sync/src/lib/business-logic.ts` L257–297 (`calculateWinnerLoserStatus`). For the readonly-role pattern, copy the existing `claude_readonly` grant style already present in `40-dashboard-views.sql` and `27b-meta-ads-breakdowns.sql`; do not look for a separate credentials doc.

Deliverables.

1. Migration `db/migrations/unified/core/004_integration_credentials.sql`: `core.integration_credentials(workspace_id uuid not null, provider text not null, account_id text not null, credential_json jsonb not null, created_at timestamptz default now(), updated_at timestamptz default now(), primary key (workspace_id, provider))`. For ecomprofits store `provider='ecomprofits'`, `account_id` = the ecomprofits account id, and `credential_json` keys `readonly_url` and `meta_account_id`.
2. Role `skynet_readonly` on ecomprofits Postgres (created out-of-band like `claude_readonly`; password to the vault) + ecomprofits PR: add `'skynet_readonly'` to the `rolname in (...)` lists at `40-dashboard-views.sql` L309 and L1519 and to `27b-meta-ads-breakdowns.sql` loops; new

declarative file `apps/web/supabase/schemas/44-skynet-read.sql` with the view below + `grant select … to skynet_readonly`; migration via `pnpm --filter web run supabase:db:diff -f skynet_read`.

3. View `public.skynet_creative_daily` (security_invoker): from `combined_meta_ad_insights i join meta_ads a on (a.meta_account_id, a.ad_id) = (i.meta_account_id, i.ad_id)`; columns `account_id, ad_account_id, ad_id, date, creative_code_raw = a.parsed_naming->>'creative_code', product_number = a.parsed_naming->>'product_number', market = a.parsed_naming->>'market', spend_native = amount_spent, spend_eur = amount_spent_eur, impressions, reach, three_second = video_plays, thruplay = video_thruplay_watched, link_clicks, purchases, purchase_value_eur, cost_per_purchase_eur, cpp_meta = cpp, breakeven_roas, seven_day_spend_eur, seven_day_purchases, seven_day_purchase_value_eur, seven_day_roas, seven_day_eligible, winner_loser_status, creative_hit, updated_at`.
4. `apps/api/core/readlayer/`: `client.py` (connection from `core.integration_credentials` where `provider='ecomprofits'`: `credential_json->>'readonly_url'`, scoping id = `account_id`), `queries.py` (`creative_daily(account_id, codes[], date_from, date_to)` — **always** `where account_id = %s`), `metrics.py` (`hook_rate = three_second/impressions`, `hold_rate = thruplay/three_second`, `roas = purchase_value_eur/spend_eur`, `cost_per_purchase_eur = spend_eur/purchases`; `None` on zero), `normalize` from S3 applied to `creative_code_raw` → `creative_code`.
5. `packages/contracts/metrics.md` (formulas, column provenance, currency rule: EUR reporting, native for display).

Interfaces. Consumed by S7 (readback), V10 (winners view).

Steps. Role + grant PR (ecomprofits, Sead owns) → view → client/queries → fixtures from a frozen sample (3 ads × 7 days, exported to `apps/api/tests/fixtures/ecomprofits_sample.json`) → staging read of real data.

Acceptance. `pytest apps/api/tests/core/test_readlayer.py` green on the fixture; PR description contains a table: 3 known ads × 1 day with hook/hold/ROAS/cost-per-purchase from this module beside the same numbers from the ecomprofits Meta dashboard (`meta-ads.tsx` holdPct, `meta-master.tsx` hookRate) — equal to 2 decimals; a query without `account_id` fails a unit test.

Out of scope. Winner/loser evaluation (S5b/S9, S3); Pinterest/Taboola; writes.

M3 · Email delivery · S · W6 · Agon · `feat/a/m3-email` (S1-rev)

Read first. ecomprofits `packages/mailers` (`MAILER_PROVIDER ∈ nodemailer|resend`, env `EMAIL_*`, `RESEND_API_KEY`).

Deliverables. `apps/api/core/mail/mailer.py` with the same provider switch; templates invite/magic/reset (sender `app@rescale.media`); staging sandbox; `MAIL_DRY_RUN=1` logs instead of sending; S1b's invites go out through it.

Acceptance. `pytest apps/api/tests/core/test_mail.py`; on staging one imported user receives the invite and signs in (screenshot).

S7 · Readback job v0 · M · `feat/s/s7-readback`

Context. Every 15 minutes, performance per creative code lands in Skynet and a lineage node is written — the last hop of the loop.

Read first. S5a module; S4 lineage API; S3 normaliser; lock doc §4.3 snapshot table.

Deliverables. Migration `005_performance_snapshots.sql`: `core.performance_snapshots(workspace_id, creative_code, ad_id, ad_account_id, date, spend_native, spend_eur, impressions, reach, three_second, thruplay, link_clicks, purchases, purchase_value_eur, cost_per_purchase_eur, hook_rate, hold_rate, roas, breakeven_roas, seven_day_spend_eur, seven_day_purchases, seven_day_purchase_value_eur, seven_day_roas, seven_day_eligible, winner_loser_status, creative_hit, source_synced_at, snapshot_at, unique (workspace_id, creative_code, ad_id, date))`. Job `apps/api/core/readback/job.py`: for each workspace with ecomprofits credentials → codes issued in the active readback window (`READBACK_WINDOW_DAYS`, default 90) (`core.creative_codes`) → `creative_daily(account_id, codes, last 14 days)` → upsert snapshots → one `performance` lineage node per (code, date) linked to the `creative_code` node. Scheduler: `apps/api/core/scheduler.py` (APScheduler or a `while` loop in the existing `scrape_worker` process — reuse `supervisord` program), interval `READBACK_INTERVAL_MIN=15`; `GET /api/performance?code=` for V10.

Acceptance. `pytest apps/api/tests/core/test_readback.py` on the S5a fixture (upsert idempotent; lineage node created once); on staging a stamped test creative's metrics appear in `core.performance_snapshots` within one cycle — post the row and the chain JSON.

Out of scope. Winner evaluation, alerts.

S10 · Creative-code loop E2E with one real ad · M · W4 · `chore/s/s10-e2e-loop` (S1-rev)

Deliverables. `docs/runbooks/LOOP_E2E.md` with proofs at each hop: (1) Production export with a code (V7) → (2) `POST /launches/:id/package` on the real Launch card of the pilot creative (S8a) → (3) the media buyer uploads the ad with the assembled ad name → (4) `select date, ad_id, nc_creative_code, amount_spent_eur, purchase_value_eur, winner_loser_status from meta_ads_dashboard_view where nc_creative_code = '<code>' order by date desc limit 20;` → (5) `core.performance_snapshots` row (S7) → (6) `GET /api/lineage/chain?type=creative_code&ref=<code>`. Mismatches fixed in the owning card the same day.

Acceptance. Runbook with all six proofs on staging with a real ad id; Agon signs the task.

S12 · Migration dry run (production PG) + rollback scripts · M · `chore/s/s12-migration-drill`

Context. Production's data (63 tables, `schema.sql`) moves from the in-container Postgres to unified PG at go-live. The drill proves it twice before it counts.

Read first. V3 spec (baseline + init guard); static-ads `docker/init-db.sh` (init guard L38, re-runs `schema.sql` every boot L72–97); `schema.sql`; implementation PRD §5 (migrate step guarantees) and §6.

Deliverables. `infra/migrate/production/`: `dump.sh` (`pg_dump -Fc` from the old host container), `restore.sh` (into unified `production` schema on staging), `checksums.py` (introspects each table's real primary-key columns from `pg_index` / `pg_attribute`, then writes `checksums.sql` with deterministic per-table queries: `count(*)` for every table, and for the 20 largest tables an ordered row hash `md5(string_agg(t::text, '|' order by <real pk columns>))`; never a literal `order by pk`), `verify.sh` (old vs new checksum table → `docs/runbooks/MIGRATION_PRODUCTION.md` with timings), `rollback.sh` (repoint `DATABASE_URL` to the old container + restore pre-migrate dump), all idempotent. Run the full drill on staging twice.

Acceptance. Runbook with two drill runs, all checksums equal, timings; rollback drill restores the pre-migrate dump under 15 min. Post the checksum table.

Out of scope. The go-live run itself (X7, Agon + Sead).

S13 · Cross-tenant isolation tests · S · W5 · `test/s/s13-isolation` (S1-rev)

Deliverables. `apps/api/tests/core/test_isolation.py`: workspaces A and B; for every route under `/api/creative-codes`, `/api/lineage`, `/api/performance`, `/api/bindings`, `/api/admin/*` and the production read routes (`GET /api/winners`, `/api/winners/{product}`, `/api/products/{product}/creatives`, `/api/products/{product}/creatives/insights`, `/api/products/{product}/sessions`, `/api/products/{product}/sessions/{id}`, `/api/video-storyboard/sessions`, `/api/video-storyboard/sessions/{id}`, `/api/localize-video/sessions`, `/api/localize-video/sessions/{id}`), user A on a B-owned id → 404/403; table in `docs/SECURITY.md`; CI required check.

Acceptance. Green in CI; 0 leaks in the table.

M5 · Retention / archive plan · S · W5 (S1-rev)

Deliverables. `docs/ops/RETENTION.md`: old static-ads host read-only for the agreed post-cutover retention window, archive point (final dump + R2 prefix manifest), deletion owner + trigger (Agon, after the S1 exit gate and the retention window); the standalone scraper and console are NOT in scope

(they stay live until S2/S3); Skynet backup retention (20 days local, Storage Box per its policy), pre-deploy dump folder policy (alert at 20 GB, never auto-pruned).

Acceptance. Doc merged; Agon and Vince acknowledged.

Lane B — Vince (Production & Brain)

WAVE	KEY	CARD	SIZE	WAITS ON	UNBLOCKS
W1	**V1**	SSO in production API/UI	S	S1a (S)	X1 (A)
W1	**V2a**	Shell skeleton	S	S1a (S)	A6 (A), V6 (V)
W1	**V3**	Production on unified Postgres	M	A1a (A)	A2 (A), A6 (A), M1 (V), S12 (S), X3 (A), X5 (A)
W2	**V4**	Model bindings + spend guard (core)	L	S1a (S), S4b (S)	S13 (S), V5 (V), V8 (V)
W3	**M1**	Static-ads per-user settings/keys migration	S	S1a (S), V3 (V)	—
W3	**V6**	Shell with links to the standalone apps	S	V2a (V)	V10 (V), V9 (V), X2 (V)
W3	**V7**	Creative code on export + lineage writes	M	S3 (S), S4 (S)	S10 (S), V10 (V)
W4	**V10**	Production polish under shell	M	V6 (V), V7 (V), S4 (S), S7 (S)	S13 (S)
W4	**V8**	Gateway client in production	M	V4 (V)	—
W5	**M2**	R2 prefix reconciliation (ADR + job)	S	S1a (S)	—
W5	**V13**	Contract/consumer tests + production module docs	S	S1a (S), S3 (S), S4 (S), S4b (S)	—
W5	**V5**	Minimal admin page: bindings + spend	S	V4 (V)	—
W5	**X2**	Upload/download through proxy	S	A3 (A), A6 (A), V6 (V)	—
S4	**V9**	Launch-board native UI	L	S8 (S), V6 (V)	S11 (S)

Static-ads facts every V card relies on: backend entry `app/backend/main.py` (routers L75–122; `auth_dep` L78, `admin_dep` L92, `intelligence_dep` L93; SPA mount L170–180); auth

`app/backend/auth.py` (`create_token` L48, `get_current_user` L61–88 reads Bearer or `?token=`); DB `app/backend/db.py` (`get_pool` L26, `get_cursor` L58); frontend `app/frontend/src/App.tsx` (routes L56–138, `RequireAuth` L47), `components/layout/{AppLayout,Sidebar}.tsx`, `lib/api.ts` (`getToken` L6 → `localStorage.auth_token`, `authHeaders` L18, `?token=` at 22 call sites incl. `getProgressStreamUrl` L1021), SSE consumers in `pages/products/SessionView.tsx:352` etc.; model calls: `app/pipeline/utils/claude_client.py` (`_call_claude_impl` L44, POST L129), `openrouter_client.py` (`generate_image` L57), `embeddings.py` (`_post_with_retry` L43), `app/backend/services/analysis_llm.py::chat` L53, `fal_client.py::_request` L198, `heygen_client.py::_request` L171, `elevenlabs_client.py` (`_with_retry` L128, `_tts_request` L483); image registry `app/pipeline/config.py` (`IMAGE_MODEL_IDS` L111, `resolve_image_model` L141, `load_settings_from_db` L189); settings `app/backend/services/settings_service.py`, `encryption.py` (Fernet, `ENCRYPTION_KEY`); export `routes/products.py:213` → `services/export_service.py::export_product` L18, zips in `routes/video_storyboard.py` L1079/L1170, client zip `lib/download.ts`; tests `pytest app/backend/tests` (no DB fixture; `conftest.py` blocks real network); Docker: `Dockerfile` L38–56 installs Postgres, `supervisord.conf` programs `postgresql` / `uvicorn` / `scrape_worker`, `docker/init-db.sh` re-runs `schema.sql` on every boot with an init guard at L38.

V1 · SSO in production API/UI · S · `feat/v/v1-sso`

Context. One cookie must open Production. Today it mints its own HS256 JWT and the frontend keeps it in localStorage and appends `?token=` to 22 image/SSE/download URLs. Rewriting those is not S1 work, so S1 keeps the token paths and swaps the token.

Read first. Lock doc §4.1 incl. "static-ads compatibility bridge"; `app/backend/auth.py` L48–88; `app/backend/main.py` L75–78; `app/frontend/src/lib/api.ts` L6–59 and the `?token=` lines; `packages/contracts/auth/verify.py` + samples (S1a).

Deliverables. (1) `auth.py`: `get_current_user` decodes with `verify_access_token` from the contracts package (cookie `rs_access` → Bearer → `?token=`, in that order); map claims → the existing user dict: `email`, `is_admin = 'admin' in roles`, `can_see_intelligence = {'researcher','admin'} & roles`, `token_version = tv`; look up `users` by email; if missing, create the row with `email`, `display_name = claim.name`, `is_admin` from roles, `token_version = tv`, and `password_hash` = a bcrypt hash of a random 32-byte secret (`users.password_hash` is NOT NULL) so legacy password login stays unusable when `LEGACY_LOGIN_DISABLED=1`. (2) Remove `create_token` usage from the API path and disable the legacy auth routes that actually exist today (`app/backend/routes/auth.py`, mounted under `/api/auth`): `POST /api/auth/login`, `PUT /api/auth/me/password`, `PUT /api/auth/users/{user_id}/password`. Keep `GET /api/auth/me`. Leave the `/api/auth/users*` admin-management routes in place until V11 replaces them. `LEGACY_LOGIN_DISABLED=1` returns 410 on each disabled route until cutover cleanup. (3) `lib/api.ts`: `getToken()` reads `localStorage.auth_token` as today; add `refreshSessionToken()` = `GET /api/auth/session-token` → `setToken()`, called on app mount (`main.tsx`) and once on any 401. V2a owns the shell login UI and route wiring.

Interfaces. Consumes S1a. Exposes nothing new.

Steps. Verifier swap → route disabling behind flag → frontend token refresh → tests.

Acceptance. `pytest app/backend/tests/test_auth_sso.py` green (valid → 200; expired / wrong-kid / revoked-tv → 401; admin route with viewer token → 403); manual on local stack: with a valid `rs_access` cookie present, open a Production page and an `<img src=…?token=…>` — both load; after `POST /api/auth/logout`, both 401 within 60 s (screenshots or curl transcript).

Out of scope. Removing `?token=` (V21, later slice); users/roles UI (V11, later slice).

V2a · Shell skeleton · S · `feat/v/v2a-shell`

Context. The unified frontend is the static-ads frontend grown into a shell. The nav becomes the product's map.

Read first. Functional design §3 (modules), §5; `App.tsx` L47–138; `Sidebar.tsx` L42–44, L107–307; `index.css` `@theme` L7–56; lock doc §4.1 cookie names.

Deliverables. (1) `src/shell/`: `ShellLayout.tsx` (top nav: Intelligence · Strategy (stub page "a later slice") · Studio · Production · Workspace · Inbox (stub) · Admin; workspace switcher reading `/api/auth/me` — consumes exactly the S1a shape `{id, email, name, ws, ws_slug, roles, permissions, feature_flags?}`), `routes.tsx` (route folders `/intelligence`, `/strategy`, `/studio`, `/production/*` = all existing routes moved under the prefix with redirects from old paths, `/workspace`, `/inbox`, `/admin`), `auth/` (`LoginPage` posting to `/api/auth/login`), `RequirePermission.tsx` (gates on `permissions[]`, never on role names). (2) Theming: keep tokens; product name appears only in the nav brand component (`BRAND_NAME` const, one place). (Invite/reset pages are delivered in V6 after S1b; the session-token refresh hook is V1's.)

Acceptance. `pnpm --filter web build` clean; `pnpm --filter web lint` clean; local: every existing Production page renders under `/production/*`; old URLs redirect; a user lacking `admin.settings.manage` cannot open `/admin` (screenshot).

Out of scope. Legacy mounts (V6), launch board (V9), admin content (V5).

V3 · Production on unified Postgres · M · `feat/v/v3-production-pg`

Context. Production's Postgres lives inside its container today (supervisord starts it; `init-db.sh` re-runs `schema.sql` every boot). On unified PG it becomes schema `production` with real migrations.

Read first. `Dockerfile` L38–56, L131–140; `supervisord.conf`; `docker/init-db.sh` L12–19, L38–97; `schema.sql` (63 `CREATE TABLE IF NOT EXISTS`, migrations block from L336 — idempotent `ALTER … IF NOT EXISTS`); `app/backend/db.py`; implementation PRD §5 (migrate step: ledger, checksum, pre-dump).

Deliverables. (1) `db/migrations/unified/production/001_baseline.sql` = `schema.sql` rewritten for schema `production` (`create schema if not exists production; set search_path = production, public;` + `create extension if not exists vector` in `public`), unchanged otherwise. (2) `apps/api/migrate.py` (the runner, shared with core): applies

`db/migrations/unified/*NNN*_.sql` in order inside one transaction each, records in `core.schema_migrations(version, checksum, applied_at)`, refuses a changed checksum, `--dry-run`, exit ≠ 0 on error; `docker compose run --rm migrate` entrypoint. (3) `db.py`: `DATABASE_URL` + `options=-c search_path=production,public`. (4) Dockerfile: remove Postgres install (L38–56), `init-db.sh`, `postgresql` program; keep `uvicorn` + `scrape_worker`. (5) `docs/MIGRATIONS.md` (forward-only, additive during S1, how to add a file). (6) Ops endpoints in `app/backend/main.py`: `GET /api/healthz` → `{"ok":true}` once the API is up and can read the migration ledger; `GET /api/version` → `{"sha":"<git sha>","migration_head":"<latest applied version>"}` (sha baked in at build via `GIT_SHA` build arg). These are the endpoints A2/A6/X3/X5 and the smoke gate use.

Acceptance. `python apps/api/migrate.py --dry-run` lists 001; `pytest app/backend/tests` green against unified PG in the existing `docker-compose.yml` dev stack (postgres service added there; A2 introduces the monorepo compose files later); full Explore + Storyboard run on the local stack (session created, images generated, storyboard delivered — screenshots); image no longer contains `postgres` binary (`docker run … which postgres` → empty); `curl -s localhost:8000/api/healthz` → `{"ok":true}` and `curl -s localhost:8000/api/version` returns the current git sha plus the latest applied migration version.

Out of scope. Data migration of the live DB (S12/X7); R2 changes.

V4 · Model bindings + usage events + spend guard · L · `feat/v/v4-bindings-spend-guard`

Context. Which model runs which job becomes a setting, not a deploy; spend gets a guard. Production is the first consumer; sidecars post usage later.

Read first. S4b contract (`packages/contracts/bindings`); `app/pipeline/config.py` L111–216 (`IMAGE_MODEL_IDS`, `CLAUDE_MODEL`, `load_settings_from_db` writing module globals); `app/backend/services/system_settings_service.py` (`resolve` L280, `set_value` L294); technical plan §2.4.4.

Deliverables. (1) Migration `core/006_bindings.sql`: `core.providers(id, workspace_id, kind check in ('openrouter','vendor','claude_subscription'), name, base_url, config_encrypted, created_at)`, `core.models(id, provider_id, slug, modality check in ('chat','image','video','tts','embedding'), price_in_usd, price_out_usd, active)`, `core.model_bindings(workspace_id, workflow_key, model_id, provider_id, updated_by, updated_at, pk(workspace_id, workflow_key))`, `core.usage_events(id bigserial, workspace_id, service, workflow_key, model, tokens_in, tokens_out, cost_usd numeric(12,6), request_id, created_at)`, `core.spend_guards(workspace_id pk, window_hours default 24, warn_usd, pause_usd, paused_at, paused_by, resumed_at)`. Seed for `rescale`: OpenRouter provider (key from env), models = today's `CLAUDE_MODEL`, `LONGFORM_OPUS_MODEL`, `HAIKU_MODEL`, `GENDER_DETECT_MODEL`, the three `IMAGE_MODEL_IDS`; bindings for workflow keys `production.concepts`, `production.longform`, `production.image`, `production.analysis`, `production.embeddings`, `production.storyboard.llm`, `production.video`, `production.tts`. (2) `apps/api/core/bindings/`: `GET /api/bindings/:workflow` (resolves binding → provider config decrypted server-side, returns the contract shape + `paused` + guard state), `POST /api/usage`

(service token or user), `PUT /api/admin/bindings/:workflow`, `PUT /api/admin/spend-guard`, `POST /api/admin/spend-guard/resume`; admin read endpoints required by V5: `GET /api/admin/bindings` (all workflow keys, current binding, available models grouped by modality, masked provider metadata) and `GET /api/admin/spend?window=24h|7d` (totals per provider and workflow from `core.usage_events` plus the current guard state); `guard.py` (rolling sum over `window_hours`; ≥ `warn_usd` → one Discord/Telegram message per hour; ≥ `pause_usd` → set `paused_at`; `SPEND_GUARD_KILL_SWITCH=1` env pauses everything). (3) `apps/api/core/gateway.py`: `@gated(workflow_key)` decorator: pre-call `resolve_binding()` → raise `SpendPaused` (HTTP 402, message "Spend guard paused this workspace at €X; an admin can resume in Admin → Spend") if paused; post-call `record_usage(tokens, cost)`; cost from `models.price_*` or the provider's usage field.

Interfaces. Exposes the contract endpoints; V8 wires production calls; V5 is the UI.

Acceptance. `pytest app/backend/tests/core/test_bindings.py` green: change the binding for `production.concepts` → next `resolve_binding` returns the new model (no restart); usage of 3 events sums correctly; crossing `pause_usd` pauses and the decorator raises 402 with the message; resume clears. Post output.

Out of scope. The admin UI (V5), wiring the call sites (V8), per-user keys (M1).

V6b · Scraper-admin SSO verifier · S · `feat/v/v6b-scraper-sso`

> **DROPPED (S1-rev): the scraper stays standalone until S2.**

V6 · Shell with links to the standalone apps · S · W3 · `feat/v/v6-shell-links` (S1-rev)

Context. Day-14 shell = Production native + links. No mounts, no SSO into the standalones.

Read first. Architecture doc §2 (boundaries), functional design §5; V2a routes.

Deliverables. Nav entries: **Intelligence** → `https://meta-ads.rescale.media` (new tab, label "opens the scraper — separate login for now"); **Studio** → the console URL (same treatment); **Workspace** → ClickUp Launches list deep link + the pilot placeholder ("native boards come with the Workspace slice"); **Strategy**, **Inbox** stubs with one sentence each; **Admin** → V5. Link targets from env (`LINK_SCRAPER_URL`, `LINK_STUDIO_URL`, `LINK_CLICKUP_LAUNCHES_URL`). `docs/INTEGRATION_CHECKLIST.md` rows: login, each nav entry, logout, one SSE stream, one upload.

Acceptance. Checklist green on staging (screenshots); `viewer` cannot open `/admin`.

Out of scope. Any native Intelligence/Studio page (S2/S3).

M1 · Per-user settings/keys migration · S · `feat/v/m1-settings-migration`

Context. Each user's encrypted OpenRouter key and generation settings must survive the move and become workspace + user scoped.

Read first. `schema.sql` L159–174 (`settings` columns incl. `api_key_encrypted`, Canva tokens), `app/backend/encryption.py`, `settings_service.py`, `config.py::load_settings_from_db` L189–216.

Deliverables. Migration `core/007_user_settings.sql`: `core.user_settings(workspace_id, user_id, key text, value_encrypted text, updated_at, pk(workspace_id,user_id,key))`; data migration in `production/002_settings_to_core.sql` (copy `settings` rows for every user → keys `openrouter_api_key`, `claude_model`, `default_aspect_ratio`, `images_to_generate`, `images_to_keep`, `canva_*`; same Fernet ciphertext, same `ENCRYPTION_KEY`); `settings_service.py` reads/writes `core.user_settings` for the current user + workspace; `load_settings_from_db(user_id)` unchanged in signature, new source. `instructions` table stays in `production` (seed for a later slice overlays).

Acceptance. `pytest app/backend/tests/test_settings_migration.py` (round-trip of a known ciphertext); on staging every imported user's settings page shows their values; a generation uses the user's key (log line masked).

Out of scope. Key rotation; per-workspace provider keys (that is V4 providers).

V7 · Creative code on export + lineage writes · M · `feat/v/v7-export-code-lineage`

Context. The stamp. Every exported asset carries a creative code from S3; the chain is written to lineage.

Read first. Lock doc §4.2 (stamping rules); S3 routes; S4 API; `routes/products.py:213` `POST /{product}/export` → `export_service.export_product` L18; `routes/video_storyboard.py` zip paths L1079/L1170; `lib/download.ts` (`downloadImagesAsZip` L17); `product_creatives.creative_code` (human-typed today, `routes/products.py:450`).

Deliverables. (1) Export flow: on export of a concept/asset, call `POST /api/creative-codes` (`kind` from asset type: image → `TM`, video → `RC`; `team_made` = session mode Explore → true, Expand → false; `product_number` from the product's naming product number — add `products.product_number` if absent, editable in product settings) → stamp: filename `<code>__<concept-slug>.<ext>`, `manifest.json {code, product_number, session_id, concept_id, naming_version:'v3', assembled_ad_name?}`, the copy-ready ad name from `POST <WORKOS_SERVICE_URL>/naming/ad-name` when S8 is live (else omitted, logged). (2) Lineage batch to S4: nodes session → concept → asset → export → creative_code, edges in order. (3) `product_creatives` upload form: `creative_code` becomes read-only, prefilled from the issued code; existing free-text codes are left as-is and reported by a new script

`scripts/report_unmatched_creative_codes.py` added in this PR (CSV output: product, creative_code, reason). (4) Client zip (`download.ts`) uses the stamped filenames from the API response.

Acceptance. `pytest app/backend/tests/test_export_code.py` (mocked S3/S4: filenames + manifest carry the code; lineage batch posted once); manual: export one concept on the local stack, unzip, show manifest; `GET /api/lineage/chain?type=creative_code&ref=<code>` returns 5 nodes (paste JSON).

Out of scope. Uploading to Meta; winner display (V10).

V8 · Gateway client in production · M · `feat/v/v8-gateway-client`

Context. Every outbound model call in Production goes through the binding + guard from V4. There are seven choke points.

Read first. V4 `gateway.py`; call sites: `claude_client.py::_call_claude_impl` L44 (workflow `production.concepts` / `production.longform` chosen by caller — add a `workflow_key` kwarg with defaults per caller module), `analysis_llm.py::chat` L53 (`production.analysis`), `openrouter_client.py::generate_image` L57 (`production.image`), `embeddings.py::_post_with_retry` L43 (`production.embeddings`), `fal_client.py::_request` L198 and `heygen_client.py::_request` L171 (`production.video`), `elevenlabs_client.py::_tts_request` L483 (`production.tts`); `render_provider.py` (`active_provider` L49) and `config.py::resolve_image_model` L141.

Deliverables. Each choke point wrapped with `@gated(workflow_key)`: model slug + base URL + key come from `resolve_binding()` (fallback to today's env only when no binding exists, with a warning log); usage posted with real token counts where the provider returns them (OpenRouter `usage`), else the model's `price_*` × estimate; `resolve_image_model` reads bindings (`production.image` default; the UI's model dropdown lists `core.models` with modality image). Remove the import-time binding of `OPENROUTER_API_KEY` in `claude_client.py` L8/L132 (late-bind).

Acceptance. `pytest app/backend/tests/test_gateway_client.py` (each choke point calls `resolve_binding` and `record_usage`; paused → 402 surfaces to the route as a clear error); staging: run one Explore session → Admin → Spend shows usage rows for `production.concepts` and `production.image` (screenshot); set `pause_usd` low → next run blocked with the message.

Out of scope. Sidecar usage posting (later slices), pricing accuracy beyond OpenRouter's usage field.

V5 · Minimal admin page: bindings + spend · S · `feat/v/v5-admin-bindings`

Read first. V4 endpoints.

Deliverables. Use V4's admin read endpoints only: `/admin/bindings` (table of workflow keys → model dropdown from `GET /api/admin/bindings`, save → `PUT /api/admin/bindings/:workflow`),

`/admin/spend` (totals per provider/workflow for 24 h / 7 d from `GET /api/admin/spend`, guard thresholds form, pause state, Resume button → `POST /api/admin/spend-guard/resume`), providers list with masked keys (last 4, from the same endpoint). Role `admin` only.

Acceptance. Screen recording: change a binding, run a job, see the new model in the usage row; trigger a pause, resume from the UI — no DB access.

Out of scope. Users/roles UI (V11, later slice), charts.

V10 · Production polish under shell · M · `feat/v/v10-production-polish`

Read first. V6, V7, S4 chain API, S7 `GET /api/performance?code=`; Appendix A.2 of the technical plan (the parity rows); pages: `WinnersGallery.tsx`, `ProductCreatives.tsx`, `SessionView.tsx`, storyboard pages, `CanvaPolishedLightbox.tsx`, localize pages, `RecreateModal.tsx`.

Deliverables. Winners gallery + product creatives show the creative code and a "Lineage" drawer (chain from S4) and the latest snapshot (S7) when present; regression pass over Explore, Expand, Storyboard (scenes/clips/narration/manifest/delivery), Canva polish + callback, Localize, bulk recreate, labs; dead nav removed; error + loading states on every `/production/*` route; `docs/checklists/production-parity.md` with these rows explicitly: Explore create/open/regenerate/download; Expand create/open/regenerate/download; Storyboard scenes/clips/narration/render/download; Canva polish callback/download; Localize create/stream/output/download; bulk recreate; labs entry pages; winners gallery + product creatives (code + lineage drawer). Each row has columns owner, expected result, actual result, screenshot link.

Acceptance. Parity checklist all green with screenshots, run on staging in the W5 parity pass.

Out of scope. Lighthouse/UX polish (V21, later slice).

V13 · Contract/consumer tests + Production module docs · S · W5 (S1-rev)

Deliverables. `apps/api/tests/contracts/` consumer tests for auth samples, bindings fixtures, creative-code fixtures, lineage fixture, the S8a launch-package fixtures (copied into `packages/contracts/launch/` by V7); CI job `contracts-consumers` required; `apps/api/production/README.md` + `apps/web/src/production/README.md`.

Acceptance. CI job green on `main`; READMEs merged.

X2 · Upload / download / SSE through the proxy · S · W5 (S1-rev)

Deliverables. Four checks on staging, in `docs/checklists/proxy-io.md`: (1) 400 MB document upload `POST /api/products/{product}/documents` via the shell; (2) storyboard clip download `GET /api/video-storyboard/sessions/{id}/scenes/{scene}/clip/{variant}` (signed R2); (3) 20-min Explore SSE `GET /api/products/{product}/sessions/{id}/progress/stream` stays open; (4) a storyboard render job survives a Caddy restart (queue leases). Pass/fail + timings.

Acceptance. All four pass on staging.

M2 · R2 prefix reconciliation (ADR) · S · `docs/v/m2-r2-prefix` (with Agon)

Read first. `app/backend/services/r2_service.py` (key construction), lock doc §4 tenancy rule, PRD §6.

Deliverables. `docs/decisions/ADR-009-r2-key-prefix.md`: all new writes use `ws/<workspace_id>/<existing key>`; existing objects keep their keys; a a later slice job copies + verifies + re-points (no deletes until verified); `r2_service.py` gains the prefix on write (`build_key(workspace_id, …)`) with a unit test; reads accept both.

Acceptance. ADR merged; `pytest app/backend/tests/test_r2_prefix.py` green (new write prefixed; old key still readable).

Lane A — Agon (Platform & Intelligence)

WAVE	KEY	CARD	SIZE	WAITS ON	UNBLOCKS
W0	**A0**	Box prep on both boxes	S	—	X4b (A), X7 (A)
W1	**A1a**	Repo + imports + CODEOWNERS	M	S0a (S)	A1b (A), A2 (A), A6 (A), S3 (S), S8 (S), V3 (V)
W1	**A1b**	CI + governance	M	A1a (A)	A2 (A), A3b (A), A6 (A)
W2	**A2**	Compose stack + local dev	M	A1a (A), A1b (A), V3 (V)	A2c (A), A3 (A), A3b (A), A4 (A), A6 (A)
W2	**A2c**	Compose resource limits + Postgres tuning + concurrency caps	S	A2 (A)	X3b (A)
W2	**A3**	Caddy for the unified app	S	A2 (A), S1a (S)	A3b (A), A6 (A), X1 (A), X2 (V)
W3	**A3b**	Deploy pipeline on the ecomprofits pattern (two targets, digest pinning, smoke gate, Telegram)	M	A1b (A), A2 (A), A3 (A)	A6 (A), X7 (A)
W3	**A6**	Staging stack on server.rescale.media	M	A1a (A), A1b (A), A2 (A), A3 (A), A3b (A), V2a (V), V3 (V), S1a (S)	S12 (S), X2 (V), X3 (A), X4 (A), X5 (A), X7 (A)
W4	**X5**	Rollback drill + cutover matrix incl. old-host cron shutdown plan	M	A6 (A), V3 (V)	M5 (S), M6 (A), S11 (S), X7 (A)
W4	**X7**	Production stack prepared on server.ecomprofits.io + go-live checklist	M	A0 (A), A3b (A), A6 (A), X5 (A)	X3 (A), X3b (A), X4b (A)
W5	**X3**	Synthetic checks + backup freshness + status page	S	A6 (A), V3 (V), X7 (A)	M4 (A)
W5	**X3b**	Supabase-protection smoke (Skynet under caps, Supabase latency unchanged)	S	A2c (A), X7 (A)	—
W5	**X4**	Backup restore drill	S	A6 (A)	—

WAVE	KEY	CARD	SIZE	WAITS ON	UNBLOCKS
W5	**X4b**	Backup path + sweep validation on the production box	S	A0 (A), X7 (A)	—
W6	**M3**	Email delivery	S	S1b (S)	—
W6	**M4**	Cross-box monitoring runbook	S	X3 (A)	—
W6	**M6**	Comms + training pack	S	X5 (A)	—
W6	**X1**	Auth/session policy tests	S	S1a (S), A3 (A), V1 (V)	—
S2	**A4**	Scraper data-access SPIKE	L	A2 (A), S1a (S)	—

Facts every A card relies on (verified against the repos): **meta-ads-scraper** — `Dockerfile` : stages `node:20-bookworm-slim` (deps, admin-build) → `mcr.microsoft.com/playwright:v1.58.1-noble` (production), PM2 via `ecosystem.config.cjs` , `EXPOSE 8080 3000` , healthcheck `curl -f localhost:8080/health` , `CMD /app/start.sh` ; `admin/middleware.ts` (cookie `admin_session` L9, redirect to `/login` L46/64, `matcher` L71); `admin/lib/auth.ts` exports `createToken/verifyToken/getSession/getCurrentUser/loginUser/getAuthCookieOptions/COOKIE_NAME` ; `server.js` L40 `API_KEY = SCRAPER_API_KEY` , auth middleware L111–115 accepts `x-api-key` or `Authorization: Bearer` ; 10 `cron.schedule` calls (L1037 */10, L1070* /15, L1157 */5, L1441* /10, L1484 */20, L1516* /30, L1674 hourly, L1691 6-hourly, L1747 quarter-hourly, L1763 daily 03:40); `package.json` scripts `test:unit = node --test test/` , admin has `build/lint` , no test; `migrations/` = 30 ad-hoc `.sql` files (no runner; `full-schema.sql` + `supabase-schema.sql` are the base); `transcriber/Dockerfile` `python:3.12-slim` , `uvicorn app:app --port 9010` ; `.env.example` keys `SUPABASE_URL/SERVICE_KEY/ANON_KEY/BUCKET, STORAGE_PROVIDER, R2_*, SCRAPING_PROVIDER, DECODO_AUTH_TOKEN, SCRAPER_API_KEY` ; live deploy = host compose at `/opt/meta-ads` (service `app` , external network `meta-ads-net` , ports 32804/32805 behind Plesk). **RescaleOS** — `console/Dockerfile` `node:22-bookworm-slim` , `WORKDIR /workspace/RescaleOS/console` , `EXPOSE 3000` , tini + `docker-entrypoint.sh` , `CMD npm start` ; `docker-compose.yml` project `name: rescaleos` , service `console` , image `ghcr.io/rescale-media/rescaleos:latest` , `env_file./console/.env.docker` , `DATA_DIR=/workspace/RescaleOS/console/data` , named volumes `rescaleos-claude-home` , `rescaleos-console-data` (never touch in Plesk GUI); `console/src/server/auth.ts` exports `SESSION_COOKIE="console_session"` , `getUserFromRequest(req)` , `requireUser/requireAdmin/requireStaff/requireStrategist(req)` , `applySessionCookie` , `checkOrigin` ; route guard = `console/src/proxy.ts` ; engine API token check in `console/src/server/scripts/engineAuth.ts` (`x-cron-token`); scripts `build/start/lint` (no test). **static-ads-automation** — top level `app/ (backend, frontend, pipeline) docker/ flowkit/ scripts/ tools/ Dockerfile supervisord.conf schema.sql docker-compose.yml` ; `Dockerfile` `node:20-alpine` (frontend build) → `python:3.12-slim` , `EXPOSE 8000` ; CI `.github/workflows/docker-build.yml` (dispatch-only). **rescale-workos** — top level `service/ (src, scripts, package.json, pnpm-lock.yaml, railway.json, nixpacks.toml, vitest.config.ts) migrations/ docs/ scripts/ presentation/ archive/ secrets/ CLAUDE.md` .

A2b · Sidecar network aliases · S · box-only

> **DROPPED (S1-rev): no sidecar mounting — the standalones are links.**

A1a · Repo + imports + CODEOWNERS · M · `chore/a/a1a-monorepo`

Context. One repo with history from four. Runtime bumps ride along so day-1 CI runs on the target versions.

Read first. Technical plan §2.1 (layout); S0a hand-over (scrubbed workos tree path + pattern file); source layouts above; scraper `Dockerfile` (Node 20 → 24: Playwright base image stays `v1.58.1-noble`, deps stages become `node:24-bookworm-slim`), static-ads `Dockerfile` (`python:3.12-slim` → `3.13-slim`; frontend stage `node:24-alpine`).

Deliverables. (1) `Rescale-Media/skynet` created; `git subtree add --prefix=<dest> <repo> main` for: static-ads-automation → `apps/api` (paths `app/backend`, `app/pipeline`, `schema.sql`, `requirements.txt`, `Dockerfile`, `supervisord.conf`, `docker/`, `scripts/`, `tools/`, `flowkit/`) and `apps/web` (`app/frontend`) — one subtree into `apps/static-ads-import/` then `git mv` into the two targets in the same PR (history preserved by `git log --follow`); meta-ads-scraper → `services/scraper`; its `transcriber/` → `services/transcriber` (`git mv`); RescaleOS → `services/agent-runtime` (`console/` + `workspaces/`, engine dirs); the scrubbed workos tree → `services/workos` (plain copy, not subtree — the old repo's history contains secrets). (2) Root `CLAUDE.md` + `AGENTS.md` (package map + the TechDesk task-discipline rules from `techdesk-skynet-integration-plan.md` §1.4), `CODEOWNERS` (`apps/api/production/ @vince`, `apps/web/ @vince`, `apps/api/core/ @sead`, `packages/contracts/ @sead`, `services/workos/ @sead`, `services/scraper/ services/transcriber/ services/agent-runtime/ infra/.github/ @agon`, `db/migrations/* @sead @agon`), `.gitignore` union + `secrets/`, `.pre-commit-config.yaml` from S0a, Renovate config. (3) Runtime bumps in the same PR: scraper Dockerfile deps stages `node:24-bookworm-slim` (rebuild `sharp`, run `node --test test/`), `apps/api/Dockerfile` `python:3.13-slim` (`pytest app/backend/tests`), frontend stage `node:24-alpine` (`npm run build`), console `node:24-bookworm-slim` (`npm run build`). (4) Old repos: `README.md` pointer + archived (GitHub "archive" flag) except rescale-workos (stays live, private, per S0a) and meta-ads-scraper/RescaleOS (stay live until their image ownership flips in a later slice — add the pointer only).

Acceptance. In the monorepo: `pytest apps/api/app/backend/tests` green on 3.13; `cd apps/web && npm run build` green on Node 24; `cd services/scraper && node --test test/` green on Node 24; `cd services/agent-runtime/console && npm run build` green; `cd services/workos/service && pnpm typecheck && pnpm test` green; `gitleaks detect` on the repo = 0; `git log --follow apps/api/app/backend/main.py | wc -l` > 1 (history kept). Paste all seven command outputs.

Out of scope. Any behaviour change; CI workflows (A1b).

A1b · CI + governance · M · `chore/a/a1b-ci`

Read first. Implementation PRD §5; existing workflows (`ecomprofits/.github/workflows/docker-build.yml` for the SSH/compose pattern, `RescaleOS/.github/workflows/docker-build.yml` for build/notify); A1a layout.

Deliverables. `.github/workflows/ci.yml` (PR + main; `dorny/paths-filter` → jobs `contracts` (`pnpm --filter contracts test` + `pytest packages/contracts`), `api` (`pytest apps/api/app/backend/tests`), `web` (`npm run lint && npm run build`), `scraper` (`node --test test/`), `runtime` (`npm run build`), `workos` (`pnpm typecheck && pnpm lint && pnpm test`), `secrets` (gitleaks with `infra/secret-scan/gitleaks.toml`, always)); `.github/workflows/deploy.yml` skeleton (build matrix → GHCR for `skynet-api`, `skynet-web` only; `deploy-staging`, `smoke-staging`, `deploy-prod` jobs stubbed with `if: vars.DEPLOY_STAGING_ENABLED == 'true'` etc.; `notify` Telegram as RescaleOS); branch protection on `main` via `gh api` script `infra/github/protect-main.sh` (required checks = the seven job names, 1 review, linear history, no force push); PR template with `Task: <key>` / `ADR: <n|none>`; `docs/CI.md`.

Acceptance. A PR touching only `apps/web` runs `web` + `secrets` + `contracts` only (screenshot of the checks list); a PR without green checks cannot merge (screenshot of the blocked merge button); `gh api repos/Rescale-Media/skynet/branches/main/protection` output pasted.

Out of scope. Real deploys (A6/X7).

A2 · Compose stack + local dev · M · `chore/a/a2-compose`

Read first. Implementation PRD §4 (services table, ports, staging isolation) and §5; V3 (`migrate` entrypoint); sidecar compose facts above.

Deliverables. `infra/compose/docker-compose.dev.yml` (project `skynet-dev`: `postgres:17` with `pg_data` volume + init SQL creating schemas `core`, `production`; `migrate` (`apps/api` image, `python apps/api/migrate.py`); `api` (uvicorn 8000, env from `infra/env/dev.env.example`); `caddy` (`infra/caddy/Caddyfile.dev`, port 38000) with exactly these dev routes: `handle /api/*` → `api:8000`; `handle` → static `apps/web/dist` with SPA fallback; unified app only, no mounted legacy UIs or sidecars); `infra/compose/compose.prod.yml` (project `skynet`: `caddy`, `api`, `postgres`, `migrate` [profile `ops`], `backup`, `synthetic`; internal network only; images by digest from `.env.release`) and `compose.staging.yml` (project `skynet-staging`, port 38001, own network and volumes, unified app only); `Makefile`: `make dev` (up + migrate + seed workspace/admin via `apps/api/core/scripts/seed_dev.py`), `make dev-down`, `make logs`; `docs/runbooks/LOCAL_DEV.md`.

Acceptance. Fresh clone: `make dev` → `docker compose -p skynet-dev ps` shows every service `healthy`; `curl -s localhost:38000/api/healthz` → `{"ok":true}`; `curl -I localhost:38000/production` returns `200`; runbook merged. Paste `ps` output.

Out of scope. Real Caddy SSO rules (A3), staging deploy (A6).

A5-lite · Agent runtime reachable + SSO · M · `feat/a/a5-runtime-sso`

> **DROPPED (S1-rev): the console stays standalone until S3.**

A3 · Caddy for the unified app · S · W2 · `feat/a/a3-caddy` (S1-rev)

Context. One gateway container for the unified app only. No `/scraper`, `/studio`, no aliases, no cross-host proxying — the standalones are links.

Read first. Architecture doc §5 (caddy row), §8 (cookies, CSRF), lock doc §4.1; `apps/web` build output.

Deliverables. `infra/caddy/Caddyfile` (env-parameterised): `:80`; `handle /api/` → *`api:8000`* (*`flush_interval -1`*, *`transport http { read_timeout 1h }`* for SSE); *`handle`* → *`apps/web/dist`* with SPA fallback; *`request_body { max_size 512MB }`*; *`header X-Request-Id {http.request.uuid}`* forwarded upstream; rate limit on *`/api/auth/`* (10/min/IP, `caddy-ratelimit` module built in `infra/caddy/Dockerfile`); trust `X-Forwarded-Proto` from `127.0.0.1` only; cookies untouched. `skynet-web` image = Caddy + built SPA.

Acceptance. `infra/caddy/test.sh` on the dev stack: `/` 200 (SPA), `/production/anything` 200 (fallback), `/api/healthz` 200, 11th `/api/auth/login` in a minute → 429, `X-Request-Id` visible in the api log, 300 MB upload passes / 600 MB → 413, `curl -N /api/products/x/sessions/y/progress/stream` streams. Paste the output.

Out of scope. TLS (Plesk); anything for the standalone apps.

A6 · Staging stack on `server.rescale.media` · M · W3 · `chore/a/a6-staging` (S1-rev)

Context. Staging is the unified app only, on the rescale box, proving migrations, the read layer, the loop and the deploy itself.

Read first. Architecture doc §6; A2 `compose.staging.yml`; A3b.

Deliverables. `/opt/skynet-staging` running project `skynet-staging` on `127.0.0.1:38001`, network `skynet-staging-net`, own volumes; `.env.staging` from `SKYNET_STAGING_ENV_B64` (read-layer role `skynet_readonly`, R2 staging prefix, Resend/SMTP sandbox, `LEGACY_LOGIN_DISABLED=0`); `vars.DEPLOY_STAGING_ENABLED=true`; nightly `pg_dump -Fc` into `/var/www/vhosts/rescale.media/private/pgdump/skynet-staging-<date>.dump` (existing sweep); seeded smoke user.

Acceptance. `https://staging.app.rescale.media/` loads with TLS; `docker compose -p skynet-staging ps` all healthy; the smoke gate passes on the next push (run link); the dump exists the next morning (`ls -l`).

Out of scope. Production (X7); any standalone app.

A4 · Scraper data-access spike → Gate G1 · L, hard stop at the end of the spike wave · `spike/a/a4-postgrest`

> **MOVED to S2 (S1-rev): "scraper data → unified PG for the Intelligence merge" spike + gate G1; text kept below for S2.**

Context. Decide whether the scraper can run against PostgREST over unified Postgres `meta_ads` with R2 media and the unified JWT — with evidence, not opinion.

Read first. Lock doc §5 (the full spec); scraper facts above (18 `.rpc()` sites / 17 functions, `STORAGE_PROVIDER=r2`, `SCRAPER_API_KEY`, `admin_session`); `migrations/full-schema.sql` + `supabase-schema.sql`; `lib/media-storage.cjs`; `admin/lib/supabase.ts`, `admin/lib/supabase-service.ts`, `server.js` L130–141 (clients with `db: { schema: 'meta_ads' }`).

Deliverables. (0) Inventory: `rg -n "\.rpc\(|\.from\(|storage\.from\(|createSignedUrl|\.channel\(|createClient\(" --glob '!node_modules' services/scraper services/transcriber` → `docs/spikes/a4-parity.csv` (file, line, kind ∈ rpc|table|storage|signed-url|realtime|client, name, key-type ∈ anon|service|user, result ∈ PASS|FAIL|DEFERRED, note); first line of the report states hit count = row count. (1) Dev: restore the scraper DB dump into unified PG as schema `meta_ads`; create roles `anon`, `authenticated`, `service_role`; run `postgrest/postgrest:v12` with `PGRST_DB_SCHEMAS=meta_ads`, `PGRST_DB_ANON_ROLE=anon`, `PGRST_JWT_SECRET=<spike HS256 secret>`; mint a service JWT (`role: service_role`) with that secret; point `SUPABASE_URL` at PostgREST and `SUPABASE_SERVICE_KEY` at the JWT. (2) Exercise every rpc (`ads_browse, ads_browse_meta, brands_browse, teardowns_browse, intel_coverage, intel_label_distributions_v2, gold_next_item, gold_sample_draw, gold_scoring_rows, mi_claim_work, refresh_brand_metrics, enqueue_brand_teardowns, enqueue_missing_classifications, enqueue_missing_transcripts, research_bump_runs_seen, research_product_inputs, research_products_ranked`) and every `.from()` path from the admin pages and the 10 crons (run each cron function once by hand: `node -e` or the existing manual endpoints) → PASS/FAIL per row. (3) Storage: `STORAGE_PROVIDER=r2` against bucket `meta-ads-media`; `admin/app/api/media/signed-url` returns a working URL; one upload + one download verified by sha256. (4) Admin auth = the V6b middleware against the unified cookie; `rg "auth\." admin/ server.js` → each hit mapped or 0. (5) Realtime: `rg "\.channel\("` → every subscription unused/polled/replaced. (6) One full cron cycle (all 10 schedules) against PostgREST with no new error lines vs a Supabase baseline log.

Acceptance. `docs/decisions/ADR-003-scraper-data-access.md` merged at the end of the spike wave with the parity table pasted, screenshot diff of the six admin pages (browse, intel, research, market-intel, gold, teardowns), the migration runbook draft (dump → restore → role map → env swap →

smoke), and the verdict: **pass** (every row PASS or DEFERRED-with-owner for provably unreachable paths; rows 3–6 PASS) → A9 cutover a later slice; **fail** → scraper keeps Supabase, A7' a later slice, the failed rows listed with a fix estimate.

Out of scope. The cutover itself; fixing the scraper beyond what the spike needs.

X3 · Synthetic checks + backup freshness + status page · S · W5 · `feat/a/x3-synthetic` (S1-rev)

Deliverables. `infra/synthetic/check.sh` (5-min cron sidecar, prod + staging): `/`, `/api/healthz`, `/api/auth/jwks`, workos bridge `/health` on Railway, one read-layer query (`GET /api/performance?code=<known>`), backup freshness (newest file in `/var/lib/skynet-backups/daily` < 24 h), host pressure (load > 24 for 10 min, available RAM < 32 GB, swap > 2 GB, disk free < 100 GB); two consecutive failures → Discord webhook + Telegram; rows in `core.synthetic_checks`; `/status` behind SSO. `docs/runbooks/ALERTS.md`.

Acceptance. `docker stop skynet-staging-api-1` → alert within 10 min (screenshots); rename last night's dump → backup-freshness alert; `/status` shows both; containers restored.

X4 · Backup restore drill · S · W5 · `chore/a/x4-restore` (S1-rev)

Deliverables. `infra/backup/restore.sh <dump> <target-container>` + `infra/backup/compare_rowcounts.py` (two Postgres URLs, `information_schema.tables`, `count(*)` per table for `core` + `production`); drill on staging: restore last night's dump into a scratch `postgres:17` container, compare, time it; `docs/runbooks/RESTORE.md`.

Acceptance. Runbook with output (counts equal, minutes); Sead repeats it from the runbook and comments "reproduced".

X5 · Rollback drill + cutover matrix · M · W4 · `chore/a/x5-rollback-cutover` (S1-rev)

Deliverables. `infra/deploy/rollback.sh <sha>` (reset checkout to `<sha>`, pin `skynet-api@<digest>` + `skynet-web@<digest>` from that run, `up -d`, restore the pre-migrate dump if the ledger head is newer) — drilled on staging; `docs/runbooks/ROLLBACK.md`; `docs/CUTOVER.md` matrix: old static-ads host → 301 to `app.rescale.media/production/*`; scraper and console URLs unchanged (links); workos untouched; DNS + TLS on the ecomprofits box; old-host crons: static-ads `scrape_worker` stopped at go-live, everything else stays; owner + trigger per row; the 14-day read-only window (M5).

Acceptance. Rollback on staging under 15 min (transcript); matrix acknowledged by Vince and Sead on the task.

X1 · Auth/session policy tests · S · W6 · `test/a/x1-auth-policy` (S1-rev)

Deliverables. `e2e/auth-policy.spec.ts` (Playwright against staging): cookie flags; refresh rotation (old → 401, reuse revokes family); logout → `/api/auth/me` 401 immediately, a static-ads `?token=` image URL → 401 within 60 s; CSRF: mutating request with `Origin: https://evil.example` → 403; `docs/AUTH_SESSION.md` final (Skynet-only scope, `?token=` bridge = ADR-005).

Acceptance. Spec green (run link); doc merged.

X7 · Production stack prepared on `server.ecomprofits.io` + go-live checklist · M · W4 · `chore/a/x7-prod` (S1-rev)

Read first. A0, A2c, A3b; architecture §5, §7, §12.

Deliverables. `SKYNET_ENV_B64` secret (prod values, escrowed); one manual `workflow_dispatch` deploy to `/opt/skynet` with `DEPLOY_PROD_ENABLED` still false in the automatic path: stack healthy on `127.0.0.1:38000`, limits visible in `docker stats`; `docs/GO_LIVE.md`: T-1 (freeze, staging smoke, S12 dry run repeated, X3b pressure test passed), T-0 (stop old `scrape_worker` → final `pg_dump` of the old static-ads DB → `migrate` → restore into `production` → checksums → `up -d` → smoke → DNS flip → old-host redirect → comms), T+1 (synthetic green 24 h, dump on the Storage Box), owners, rollback trigger.

Acceptance. `docker compose -p skynet ps` all healthy on the production box; `curl --resolve app.rescale.media:443:<ip> https://app.rescale.media/api/healthz` → ok; checklist signed by Agon + Sead.

M6 · Comms + training pack · S · W6 (S1-rev)

Deliverables. `docs/comms/` one page per audience: Bian (nothing changes for RescaleOS; the console stays where it is; Studio merge comes with S3 and needs his agreement), Cyrus + creative (Production lives at `app.rescale.media`, one login, exports carry the creative code, the package appears on the Launch card), media buyers (nothing changes in ClickUp; the package on the card), ops/research (scraper unchanged, reachable from the shell link), leadership (S1 exit gate met, the slice sequence). Sessions booked.

Acceptance. Pages merged; invites sent (screenshot).

A8 · Scraper ops soak (passive) · S · reviewed at the end of S1

> **DROPPED (S1-rev): the scraper is not part of the Skynet S1 runtime. Standalone scraper changes are allowed only through bridge track B.**

M4 · Cross-box monitoring runbook · S · W6 (S1-rev)

Deliverables. Extend `infra/synthetic/check.sh` with: old static-ads host (301 after go-live), `meta-ads.rescale.media` and the console URL (200 — they stay live as standalones), Railway workos `/health`; `docs/runbooks/LEGACY_HOSTS.md`: what runs where (production box vs rescale box vs Railway), who to call, which crons stay.

Acceptance. Rows on `/status`; runbook merged.

A0 · Box prep on both boxes · S · W0 · box-only (S1-rev)

Context. Production runs on `server.ecomprofits.io` (48 cores / 250 GB / 1.2 TB free, Plesk + Portainer, production Supabase on the same box); staging on `server.rescale.media`. Everything the deploy pipeline assumes about the boxes is created here, once, by hand.

Read first. Architecture doc §5, §6, §9, §10; ecomprofits `.github/workflows/docker-build.yml` (the SSH/compose pattern already used on the production box); `project_rescale_plesk_server` notes (never edit containers in the Plesk GUI; Portainer = viewer).

Deliverables. On `server.ecomprofits.io`: `mkdir -p /opt/skynet` (git clone of the monorepo, branch `main`, deploy key read-only); `docker login ghcr.io` with a `read:packages` PAT; `mkdir -p /var/lib/skynet-backups/{daily,pre-deploy}` owned by the backup account, mode 750; Plesk vhost `app.rescale.media` → proxy `127.0.0.1:38000` with Let's Encrypt, HSTS, `client_max_body_size 512m` (DNS still points at the old host until go-live — issue the certificate via the staged hostname or DNS challenge); a Plesk Scheduled Task (daily 02:30) syncing `/var/lib/skynet-backups/daily` to the Hetzner Storage Box prefix `skynet/`; record a baseline `docs/ops/box-baseline-ecomprofits.md` (`nproc`, `free -g`, `df -h`, `docker stats --no-stream`, `uptime`). On `server.rescale.media`: `/opt/skynet-staging` clone, GHCR login, vhost `staging.app.rescale.media` → `127.0.0.1:38001` with TLS, staging dumps go to the existing Plesk `private/pgdump` sweep. Both: an SSH deploy user with the GitHub secrets `SERVER_HOST_PROD/STAGING`, `SERVER_USER`, `SERVER_SSH_KEY`, `SERVER_DEPLOY_PATH_PROD/STAGING` set.

Acceptance. Post in the task: `ls -ld /opt/skynet /var/lib/skynet-backups/*` on prod; `curl -I https://app.rescale.media` (with `--resolve` to the box) → 502 from Plesk (vhost exists, nothing behind it yet) and `curl -I https://staging.app.rescale.media` → same on staging; `docker pull ghcr.io/rescale-media/ecomprofits:latest` succeeds on both (login works); the baseline doc merged.

Out of scope. Any container; DNS flip (X7).

A2c · Compose resource limits + Postgres tuning + concurrency caps · S · W2 · `chore/a/a2c-limits` (S1-rev)

Context. Skynet is a bounded tenant on the production Supabase box: ≤ 12 cores / ≤ 32 GB in total, enforceable in host-side compose.

Read first. Architecture doc §9 (the table and thresholds — this card implements it exactly); Docker compose fields `cpus`, `mem_limit`, `mem_reservation`, `pids_limit`, `restart` (not `deploy.resources`).

Deliverables. In `infra/compose/compose.prod.yml` (and the same values in `compose.staging.yml`): api `cpus: 6`, `mem_limit: 12g`, `mem_reservation: 8g`, `pids_limit: 512`; postgres `cpus: 4`, `mem_limit: 12g`, `mem_reservation: 8g`, `pids_limit: 256`, command flags `-c shared_buffers=3GB -c effective_cache_size=8GB -c work_mem=16MB -c maintenance_work_mem=512MB -c max_connections=80 -c max_wal_size=4GB`; caddy `cpus: 1`, `mem_limit: 1g`; backup and synthetic `cpus: 0.5`, `mem_limit: 512m`, backup wrapped in `nice -n 10 ionice -c2 -n7`; every service `restart: unless-stopped`. App env: `MAX_CONCURRENT_IMAGE_JOBS=4`, `MAX_CONCURRENT_STORYBOARD_JOBS=2`, `MAX_CONCURRENT_EXPORT_JOBS=4`, enforced by the job worker's semaphores (`apps/api/core/jobs/limits.py`) — no unbounded thread fan-out remains in `pipeline_runner.py` / `storyboard_queue.py`. `docs/ops/RESOURCE_LIMITS.md` = the table + how to change it.

Acceptance. `docker compose -f infra/compose/compose.prod.yml config` shows the limits; on the dev stack `docker stats` shows the caps; `pytest apps/api/tests/core/test_job_limits.py` (a 10-job burst never exceeds 4 concurrent image jobs); `SHOW shared_buffers` on the stack's Postgres = 3GB.

Out of scope. Limits for scraper/runtime services (S2 re-issue).

A3b · Deploy pipeline on the ecomprofits pattern · M · W3 · `chore/a/a3b-deploy` (S1-rev)

Context. Push to `main` → build → deploy staging (`server.rescale.media`) → smoke gate → deploy prod (`server.ecomprofits.io`) → Telegram. Same pattern ecomprofits uses on the production box, with digest pinning and a whole-file env secret.

Read first. Architecture doc §7 (the exact steps, guarantees, rollback); `ecomprofits/.github/workflows/docker-build.yml` (SSH via `appleboy/ssh-action`, compose pull/up, Telegram curl format); RescaleOS `DEPLOY.md` §2b (env-from-secret validation, `.bak-*`); A1b's `deploy.yml` skeleton.

Deliverables. `.github/workflows/deploy.yml`: `build` (GHCR `skynet-api`, `skynet-web`, tags `main-<sha>` + `latest`, outputs `API_DIGEST`, `WEB_DIGEST`, `GIT_SHA` build arg) → `deploy-staging`

(SSH; `git fetch && git reset --hard origin/main`; decode `SKYNET_STAGING_ENV_B64`, validate (non-empty, required keys list in `infra/deploy/required-keys.txt`, no `ANTHROPIC_API_KEY=`), `.env.staging.bak-<ts>`, atomic write; write `.env.release` with `IMAGE_TAG`, `API_IMAGE_DIGEST`, `WEB_IMAGE_DIGEST`; `docker compose run --rm migrate`; `pull`; `up -d`; wait `/api/healthz` 60 s) → `smoke-staging` (`infra/smoke/run.sh`, architecture §7 step 3) → `deploy-prod` (same steps on `/opt/skynet` with `SKYNET_ENV_B64`, the **same digests**, gated by `vars.DEPLOY_PROD_ENABLED`; pre-migrate `pg_dump -Fc` to `/var/lib/skynet-backups/pre-deploy/`; abort if `df --output=avail /` < 20 GB) → `notify` (Telegram, ecomprofits format). Compose files reference `${API_IMAGE_DIGEST}` / `${WEB_IMAGE_DIGEST}` (`image: ghcr.io/rescale-media/skynet-api@${API_IMAGE_DIGEST}`), never `latest`. `infra/deploy/rollback.sh <sha>` per architecture §7. `docs/runbooks/DEPLOY.md`.

Acceptance. A push to `main` runs green through `smoke-staging` with `deploy-prod` skipped (flag off) — run link; `.env.release` on staging shows the digests from that run's build job; `rollback.sh <previous-sha>` on staging brings back the previous `/api/version` sha in under 15 min (transcript).

Out of scope. Production enablement (X7).

S8a · Narrow workos bridge: ad-name assembly + attach export package · M · W3 · old workos repo, branch `feat/skynet-bridge-s1`, deployed to Railway (S1-rev)

Context. S1 needs two things from workos: the assembled v3 ad name for an export, and the export package attached to the **already-existing** Launch card. No launch list/read/create/update API — that is S8b (S4).

Read first. Lock doc §4.5 (human-owned vs machine-owned fields); workos `service/src/app.ts`, `index.ts` (route mounting before `serve`), `modules/cascade/webhooks.ts` (`registerWebhookRoute` shape), `modules/naming/assemble.ts` (`assembleAdName`, `AD_REQUIRED`), `clients/clickup.ts` (`resolveField`, `setCustomField`, `getTask`, comments), `config.ts` (`EnvSchema`), test pattern `apply-names.test.ts`.

Deliverables. `service/src/modules/skynet-bridge/` with `registerSkynetBridgeRoutes(app, deps)` mounted in `index.ts`; auth = Skynet service token (RS256, `typ=service`, verified with `SKYNET_JWT_PUBLIC_KEY_PEM`, `jose`). Endpoints: `POST /naming/ad-name` (`AdNameInput` → `assembleAdName` → `{ad_name}` or `{withheld_reason, missing[]}`); `POST /launches/:id/package` (body `{creative_code, ad_name, assets:[{url,kind,filename}], manifest_url, source:"skynet"}`; verifies the Launch task exists and its linked Creative's code equals `creative_code`; writes only `Creative URL` (durable app URL) if empty, posts one comment "Skynet export package" with the asset links + manifest, adds the `skynet-package` tag; never touches status, assignee or any machine-owned field; idempotent on `(task_id, creative_code)` via a comment marker). Config: `SKYNET_JWT_PUBLIC_KEY_PEM`, `SKYNET_BRIDGE_ENABLED` (default false). Fixtures in `service/src/modules/skynet-bridge/fixtures/` (copied into monorepo `packages/contracts/launch/` by V7).

Acceptance. `pnpm vitest run src/modules/skynet-bridge` green (mocked ClickUp: ad name assembled for a complete input, withheld with `missing[]` for an incomplete one; package attached

once, second call no-op; wrong code → 409; bad token → 401); on a sandbox Launch task: curl both endpoints, screenshot the comment + `Creative URL`. Deployed to Railway with the flag on.

Out of scope. Creating launches; status changes; the native board (S8b/V9, S4).

X3b · Supabase-protection smoke · S · W5 (S1-rev)

Context. Prove Skynet under its caps cannot degrade the production Supabase.

Deliverables. `infra/smoke/pressure.sh`: on the production box (before DNS flip, X7 stack up): run 4 concurrent Explore sessions + 2 storyboard renders + a 400 MB upload against `127.0.0.1:38000` for 20 min while sampling every 30 s: `docker stats` for `skynet-*` and `supabase-db/pooler/kong`, `uptime`, and a `pg_stat` latency probe against the ecomprofits pooler (`select 1` p95 via `pgbench -S` on the read-only role, 1 client). Report in `docs/ops/pressure-test-<date>.md`.

Acceptance. Skynet containers never exceed their caps (no OOM, no throttling beyond `cpus`); Supabase p95 latency during the run within 10 % of the 10-min baseline before it; host load < 24. If not, A2c limits are lowered and the test repeated — no go-live before it passes.

X4b · Backup path + sweep validation on the production box · S · W5 (S1-rev)

Read first. Architecture doc §10; A0 (folders, Scheduled Task).

Deliverables. With the X7 stack up on `server.ecomprofits.io`: the `backup` sidecar writes `/var/lib/skynet-backups/daily/skynet-<date>.dump` (mode 640, owner = backup account); the Plesk Scheduled Task runs and the file appears on the Storage Box under `skynet/` (`ls` over SSH port 23); retention 20 days enforced by the sidecar (`find -mtime +20 -delete` limited to that folder); a pre-deploy dump lands in `pre-deploy/` on the first prod deploy; `X3` freshness check green. `docs/runbooks/BACKUP.md`.

Acceptance. Screenshot/ls of the file on the Storage Box; freshness check green; permissions listed.

Slices S2–S5 — executable specs

Same standard as S1. Waves are per slice (`S2-W1` …). A slice's W0 cards depend on the previous slice's exit gate (delivery plan §2), not on a card. Discovery (D) cards are real work: an interview or confirmation with a named role, a written artefact, and a sign-off comment; their output is the input of the build cards that list them.

Discovery cards (all slices)

D21 · Research operator workflow confirmation · A · S · S2-W0

Context. W1–W3 in the workflow map are `mixed`: the tooling is code-backed, the operator rules are not. S2 builds the native Intelligence pages on those rules.

Deliverables. `docs/discovery/s2-operator-rules.md`: what makes a competitor worth tracking; cadence per brand/lens; keep/discard rules for ads and families; when a finding becomes a brief; the evidence required before a product is promoted; who seeds research keywords; who approves Promote to product; the rework loop when a ranked opportunity is rejected or deferred; where the loop breaks today. Interview: Agon + the weekly research operator (45 min), plus one screen-share of a real weekly pass.

Acceptance. Doc merged; Agon and the operator comment "confirmed" on the task; A13/A14/A15 cite the rules they implement.

D22 · Product bridge field confirmation · S · S · S2-W0

Deliverables. `docs/discovery/s2-product-bridge-fields.md` with Sven + Cyrus: the exact ClickUp Product card fields (`LISTS.products`: Product ID = naming product number, market economics, status) that Promote to product must create/link, the mapping to the static-ads product model (`routes/products.py:43` product, `:151` brand identity, `:248` looks, `:282` documents), and who owns each field afterwards. Table: source · target · owner.

Acceptance. Doc merged; Sven and Cyrus confirmed; A15 cites it.

D23 · Reddit VOC seed confirmation · A · S · S2-W0

Deliverables. `docs/discovery/s2-reddit-voc-seed.md` with Mathew: the first keyword groups (cravings, blood sugar — confirm wording and subreddits/search scopes), collection cadence, retention (never truncated per group), export columns (post/comment id, subreddit, date, score, text, group, keyword hit), who consumes the export.

Acceptance. Doc merged with Mathew's confirmation; A18 cites it.

D23b · Evidence lens seed confirmation · A · S · S2-W0

Deliverables. `docs/discovery/s2-evidence-lens-seed.md` with Mathew: for each VOC keyword group (first: cravings, blood sugar) the six-lens terms — mechanism, concern, ingredient(s), common solutions — plus the first RescaleOS product the evidence layer is proved on; collection cadence and the monthly re-walk; export columns (PMID, title, journal, year, evidence class, lens, group, plain-language finding, citation); who consumes the export; the claims-policy sign-off (evidence class never authorizes disease copy).

Acceptance. Doc merged with Mathew's confirmation; A18b cites it.

D31 · Bian coexistence + migration-window agreement · A · S · S3-W0

Context. RescaleOS is Bian's live production loop (NPT/BPT). S3 moves the engine, the loops and the skills onto the stack. Nothing starts without his explicit agreement.

Deliverables. `docs/decisions/D31-bian-coexistence.md`: the cutover order (runtime on Postgres → loops on the stack → skills in-app → console retired), the coexistence period (both run, which is

authoritative), the migration window, the failure path (back to the Mac), what must never be automated, and what changes for Bian day to day.

Acceptance. Signed (comment) by Bian and Agon; A10/A11/A20 cite it.

D32 · Hook + strategy expectations · V · S · S3-W0

Deliverables. `docs/discovery/s3-hook-and-strategy.md` with Mathew + Cyrus: hook-session gate semantics (who approves, evidence packet, per-asset vs per-batch), what makes the coverage map actionable (which decisions, which existing artefacts they trust, what stays human), the brand-tone band and claim-validation rules for hook seeds; the exact rule for "validated learning" vs "personal taste" and what qualifies as a human-rated output for shared memory; and the six accelerator questions (`claude-code-accelerator-digest.md` §6): which three artefacts he touches daily; what crosses from personal instinct into shared memory and who approves; losers/dead angles first-class or implied; swipe bank shared or personal; whether "third time = workflow" deserves a productised capture flow; whether "one format first, LFS, blood sugar" is a pilot tactic or the general rule.

Acceptance. Doc merged with both confirmations; V14/V16/V18 cite it.

D33 · Script-card bridge + demand semantics · S · S · S3-W0

Deliverables. `docs/discovery/s3-script-bridge.md` with Sven + the ops owner: the Script card lifecycle (statuses, who flips), Demand Line provenance rules (`demand-week.ts` , `spawn.ts`), QA-gate ownership (`qa-gates/*`), what the bridge must preserve while ClickUp owns the Scripts list, the failure rules; Friday-call outcome ownership, manual exception paths, and the workaround rules that must survive until ClickUp is read-only.

Acceptance. Doc merged; S16 cites the state map.

D41 · Launch pilot confirmation · S · S · S4-W0

Deliverables. `docs/discovery/s4-launch-pilot-confirmation.md` with Cyrus + one active media buyer: the exact launch package, the proof-of-launch loop, which ClickUp fields they actually read, what they still rewrite by hand, what delays launches.

Acceptance. Doc merged; S6 and S8b cite it.

D42 · Board authority + mirror-failure confirmation · S · S · S4-W0

Deliverables. `docs/discovery/s4-board-authority.md` with Sven + the ops owner: authoritative states per list (from the status templates in `fields-manifest.json`), the field definition for `ugcCreators` (none exists today), acceptable mirror lag, failure handling (`reconcile-cron.ts` , `qa-gates/*`), machine- vs human-owned fields per list (from `domain/ids.ts`), and the ordered list of the remaining 12 boards after the launch board.

Acceptance. Doc merged with the ordered board list; S6, S26, S18, S19 cite it.

D51 · Billing + retention confirmation · S · S · S5-W0

Deliverables. `docs/discovery/s5-billing-retention.md` with the ops owner: the manual-invoicing process, meter granularity (model calls, scraper/runtime jobs, bridge operations), invoice-export shape, retention windows per data class, deletion approvals, who may request export/deletion.

Acceptance. Doc merged; S24, S25 cite it.

D52 · Roles + onboarding confirmation · V · S · S5-W0

Deliverables. `docs/discovery/s5-roles-onboarding.md` with Sven + Mathew: role matrix across Production, Intelligence, Studio, Workspace; admin boundaries; workspace-creation flow (Rescale admin only); invite rules.

Acceptance. Doc merged; S21, S23, V21 cite it.

D53 · Security review scope confirmation · A · S · S5-W0

Deliverables. `docs/discovery/s5-security-scope.md`: review scope (tenancy, credentials, deletion safety, abuse cases, load envelopes), external vs internal reviewer, sign-off format.

Acceptance. Doc merged; A17, S20 cite it.

S5 — SaaS hardening (specs)

S21 · RLS rollout + tenancy CI for core, production, meta_ads, studio · S · L · S5-W1

Context. Every table already carries `workspace_id` (S1 rule). S5 turns that into enforced isolation at the database.

Read first. `db/migrations/unified/*` (all schemas), `apps/api/core/auth/deps.py` (how the request's workspace is known), `apps/api/tests/core/test_isolation.py` (S13), Postgres RLS docs; the ecomprofits pattern `has_role_on_account()` in `apps/web/supabase/schemas/03-accounts.sql` as prior art.

Deliverables. Migration `core/0NN_rls.sql` + one per schema: `alter table … enable row level security`, policies `workspace_isolation` using `current_setting('app.workspace_id')::uuid`; the API sets `set local app.workspace_id` per request/transaction (`apps/api/core/db/session.py`), jobs set it per workspace loop; a `bypass_rls` role only for migrate/backup; `pytest apps/api/tests/security/test_rls.py` (cross-workspace fixtures for all four schemas, direct SQL as the app role → 0 rows); CI required.

Acceptance. Tests green; `select count(*) from production.sessions` as the app role with a foreign workspace id = 0; existing S13 tests still green.

Out of scope. Billing; UI.

S22 · Per-workspace credentials, keys, quotas + admin APIs · S · L · S5–W2

Read first. `core.integration_credentials` (S5a), `core.providers` (V4), `core.spend_guards`; D51.

Deliverables. `core.workspace_quotas(workspace_id, key, limit, window, action ∈ warn|pause)` for model spend, scraper jobs, runtime sessions, storage GB; provider keys and residential/R2 settings per workspace (encrypted); admin APIs `GET/PUT /api/admin/workspaces/:id/credentials`, `/quotas`; resolver `resolve_credentials(workspace, provider)` used by every caller; `pytest apps/api/tests/core/test_workspace_credentials_and_quotas.py`.

Acceptance. Tests green; a workspace with no key of its own cannot use the platform default unless flagged `inherit_platform_keys`.

A22 · Sidecar + runtime quota enforcement · A · M · S5–W2

Read first. S22 resolver; scraper `settings` reads, transcriber, agent runtime `accounts.ts`, workos config, production `gateway.py` (V8).

Deliverables. Every service resolves workspace-scoped credentials and checks quotas before work: scraper crons per workspace, transcriber batches, runtime sessions, workos bridge calls, production model calls; pause → 402/skip with a logged reason; `pnpm --filter scraper test && pnpm --filter agent-runtime test` extended; a staged quota-breach transcript for two services.

Acceptance. Transcript attached; no service reads a global key when a workspace key exists.

S23 · Audit log + admin provenance · S · M · S5–W3

Context. Multi-tenant operation needs an unforgeable record of who did what in which workspace; S1–S4 only log locally.

Deliverables. `core.audit_log(id, workspace_id, actor_id, actor_kind ∈ user|service|system, action, target_type, target_id, evidence jsonb, created_at)`; writers for admin actions, gate decisions, bridge mutations, membership changes, impersonation (if any), and the role lifecycle (`role.created`, `role.updated`, `role.deleted`, `member.role_granted`, `member.role_revoked`) with `evidence` holding the previous and new role definition and the previous and new effective permissions; `GET /api/admin/audit?…`; retention per D51; `pytest apps/api/tests/security/test_audit_log.py` with an end-to-end chain for a role change.

Acceptance. Tests green; the chain artefact attached.

Out of scope. Impersonation features themselves; external SIEM export.

V21 · Admin / users / roles parity + invite-only onboarding + workspace creation · V · L · S5-W3

Read first. D52; V5 admin; S1b invites; S22/S23 APIs.

Deliverables. Admin area: workspaces (create — Rescale admin only — with name, slug, plan notes); members screen with **multi-role assignment**; **role editor** (admin only): system roles read-only, custom roles per workspace — create / clone a system role / edit permissions / delete (blocked while assigned); effective-permissions drawer per member; no direct per-user permission editing; invites assign one or more roles at invite time; credentials + quotas per workspace (masked), audit viewer, invoice exports (S24). Onboarding: invite → accept → first login → workspace switcher. `pnpm --filter web test -- admin-users-roles-onboarding`.

Acceptance. Walkthrough: create workspace → invite user → assign roles → user sees only their workspace; recording attached.

S24 · Usage metering ledger + invoice export · S · L · S5-W4

Context. Manual invoicing is in scope, so usage must be metered in a stable ledger and exported as invoice-ready monthly totals without Stripe self-serve flows.

Read first. `core.usage_events` (V4), scraper/runtime job tables, D51.

Deliverables. `core.usage_ledger` rollups per workspace per day (model spend, scraper jobs, transcriber minutes, runtime sessions, storage GB, bridge ops); `GET /api/admin/usage?workspace&month`; export CSV/JSON per workspace-month for manual invoicing; `pytest apps/api/tests/billing/test_usage_metering.py`. No Stripe.

Acceptance. Tests green; one export for a fixture month attached.

Out of scope. Payment collection, subscription checkout, customer-facing billing portals, plan-pricing automation.

A17 · Hardening: load tests, abuse cases, security-review remediation · A · L · S5-W4

Deliverables. Load test of the full stack under S5 quotas (`infra/smoke/load.sh`); abuse cases (cross-workspace ids in every mutating route, token replay, oversized uploads, runaway jobs); dependency + image scanning in CI; secrets audit; findings → fixes; `docs/security/S5-security-review.md` (findings, status, evidence).

Acceptance. Review doc merged with every finding closed or explicitly accepted by Agon; load summary attached.

S25a · Workspace export + retention jobs · S · M · S5-W5

Context. Tenants need their data back on request and stale data must age out by policy before deletion is meaningful.

Deliverables. Workspace export (all schemas + R2 prefix manifest) as a signed download; retention jobs per data class (D51) with audit entries; `pytest apps/api/tests/security/test_export_retention.py`.

Acceptance. Test green; one export bundle for a fixture workspace attached.

Out of scope. Deletion itself (S25b).

S25b · Deletion queue, dry-run, purge across Postgres / R2 / lineage · S · M · S5-W6

Context. Deletion must be complete across Postgres, R2 and lineage, approved, and provable — the last SaaS-hardening control.

Deliverables. Deletion request queue with approval, legal-hold override, dry-run report, purge across every schema + R2 prefixes + lineage nodes, audit entries; `pytest apps/api/tests/security/test_delete_purge.py`.

Acceptance. Test green; one dry-run deletion report attached.

Out of scope. Legal review of retention law; backups purge (handled by retention policy on the Storage Box).

S20 · SaaS readiness review → 100 % · S · S · S5-W6

Context. The single go/no-go that defines 100 %: every control above verified, every workflow native, nothing pending.

Deliverables. `docs/decisions/S20-saas-readiness-review.md`: checklist over tenancy (S21), credentials/quotas (S22, A22), audit (S23), admin/onboarding (V21), metering/invoicing (S24), hardening (A17), retention/deletion (S25); every workflow in the source map native; standalones retired; company-ops lists still in ClickUp by design.

Acceptance. Review merged with pass on every line and an empty follow-up list — the definition of delivered.

Out of scope. New scope; anything not already carded.

S2 — Intelligence merge (specs)

Facts every S2 card relies on (verified against `meta-ads-scraper`): one container, `ecosystem.config.cjs` runs `server.js` (Express :8080, all `node-cron` jobs) + `admin/` (Next.js 14, :3000); DB = Supabase schema `meta_ads` via `supabase-js` clients with `db: { schema }` (`server.js:130–141`, `admin/lib/supabase.ts`, `admin/lib/supabase-service.ts`); 18 `.rpc()` sites / 17 functions (`ads_browse, ads_browse_meta, brands_browse, teardowns_browse, intel_coverage, intel_label_distributions_v2, gold_next_item, gold_sample_draw, gold_scoring_rows, mi_claim_work, refresh_brand_metrics, enqueue_brand_teardowns, enqueue_missing_classifications, enqueue_missing_transcripts, research_bump_runs_seen, research_product_inputs, research_products_ranked`); tables `brands, ads, ad_tags, tags, ad_classifications, ad_transcripts, ad_teardowns, ad_rank_snapshots, admin_users, brand_metrics, brand_label_shares, scrape_jobs, scrape_job_ads, scrape_job_windows, settings, proxies, creative_gold_labels, gold_evaluations, gold_sample_batches, gold_sample_items, mi_scopes, mi_documents, mi_document_scopes, mi_brand_aliases, mi_business_snapshots, mi_fetch_queue, research_niches, research_keywords, research_runs, research_run_queries, research_products, research_product_ads, research_product_scores`; media `lib/media-storage.cjs` (`STORAGE_PROVIDER=r2`, bucket `meta-ads-media`, signed URLs via `admin/app/api/media/signed-url`); crons in `server.js`: scheduler *`/30`*, *reaper* `/10`, brand metrics *`/20`*, *transcription* `/10`, classification *`/15`*, *teardown* `/5`, Reddit `5,20,35,50`, Trustpilot hourly, Amazon 6-hourly, retention `40 3`; *engines* *`scraper.js`* *(Playwright quick scrape),* *`lib/deep-crawl/`*, `lib/research/{runner,cluster,score}.cjs`, `lib/teardown/{worker,prompts}.cjs`, `lib/market-intel/`, *`lib/analysis/{worker,taxonomy,gold-eval}.cjs`*, *`lib/creative-family.cjs`*, *`lib/transcription/`*; admin pages (19): `/` (brands index), `/brands/[brandId]`, `/brands/new`, `/ads`, `/breakdowns`, `/intel`, `/intel/gold`, `/rank`, `/research`, `/research/keywords`, `/runs`, `/market-intel`, `/market-intel/{amazon,trustpilot,reddit,scopes}`, `/settings`, `/users`, `/login`; 40 admin API routes under `admin/app/api/` *(ads tags, auth, brands + schedule + teardowns, crawl, gold, intel, jobs + retry, market-intel aliases/documents/thread/reddit-stats/scopes/run + amazon/trustpilot reviews/run/scopes/stats, media/signed-url, rank + history, research + keywords, scrape, settings, tags, teardowns, users); Express routes in* *`server.js`*: *`/health`*, *`/jobs`*, `/scrape`, `/crawl`, `/jobs/:id/cancel`, `/transcripts/{run,stats}`, `/analysis/{run,stats}`, `/teardowns/`, */gold/batches*, `/research/*`, `/market-intel/{trustpilot,amazon,ingest}/{run,stats}`; env: `PORT, SCRAPER_API_KEY, SUPABASE_URL/ANON_KEY/SERVICE_KEY/BUCKET, STORAGE_PROVIDER, R2_ACCOUNT_ID/ACCESS_KEY_ID/SECRET_ACCESS_KEY/BUCKET_NAME/PUBLIC_URL, TRANSCRIBER_URL, DEFAULT_PROXY_GEO, MARKET_INTEL_ENABLED, MARKET_INTEL_AMAZON_ENABLED, MARKET_INTEL_TRUSTPILOT_ENABLED, MARKET_INTEL_AUTHOR_SALT`; RPC signatures (for the `pg` layer): `ads_browse(17 filter args, sort_key_arg text='newest', limit_arg int=144, offset_arg int=0)` → 33-col table, `ads_browse_meta(same 17)` → `(total_count, undated_in_scope)`, `brands_browse(15 args)` → 28-col table, `teardowns_browse(brand, status, version, limit=25, offset=0)`, `enqueue_brand_teardowns(brand, version, limit=50, requested_by)` → int, `enqueue_missing_transcripts(batch_limit=500)`, `enqueue_missing_classifications(version, batch_limit=500)`, `current_analysis_version()`, `refresh_brand_metrics()`, `gold_sample_draw(batch)`, `gold_next_item(batch, labeller, lease_minutes=30)`, `gold_scoring_rows(batch)`, `intel_label_distributions(version)`, `intel_label_distributions_v2(version, brand_ids[], start, end, media_type, min_confidence, include_fallback=true)`, `intel_coverage(same 7)`, `mi_claim_work(kind, limit=10, lease_seconds=900)` → `setof mi_fetch_queue`, `research_bump_runs_seen(run)`,

`research_product_inputs(run)`, `research_products_ranked(run, niche, trend, min_score, search, limit=100, offset=0)` → 25-col table; triggers `gold_release_claim`, `preserve_ad_owner_brand`, `set_updated_at`; helpers `parse_ad_date`, `progress_heartbeat`; static-ads' duplicate scraper `app/backend/routes/scraped_ads.py`, `services/decodo_client.py`, `scrape_runner.py`, `ad_collections`.

A4 · Intelligence data-path proof + pressure test → ADR-003 · A · L · S2-W1

Context. Decided: real port to direct Postgres, no PostgREST. A4 proves it before A9 commits — schema, RPC semantics, load on the shared box.

Read first. `migrations/full-schema.sql`, `supabase-schema.sql`, every `migrations/*.sql` *`create function meta_ads.`*; `server.js` client creation and every `.rpc(` / `.from(` / `storage.from(` site (`rg` inventory, one row each); architecture §9; D21–D23.

Deliverables. (1) Restore a fresh scraper dump into a scratch Postgres 17 as schema `meta_ads` (roles `anon/authenticated/service_role` dropped; one app role); run every RPC by hand with fixture args and record result parity vs Supabase (`docs/spikes/a4-parity.csv`: rpc/table/storage/realtime row, result). (2) Prototype the `pg` client layer (`services/scraper/lib/db.cjs`: pool, `rpc(name, args)` → `select * from meta_ads.<fn>(...)`, `from(table)` helpers used by the code today) on three representative paths (browse, enqueue teardown, research ranking). (3) Pressure test on the production box with the engine + transcriber under proposed caps (8 cores / 24 GB) while sampling Supabase latency (X3b method). (4) `docs/decisions/ADR-003-intelligence-data-path.md`: direct `pg` is the path; limits table for A2d; the migration runbook draft.

Acceptance. Parity CSV with every row PASS or explained; pressure report shows Supabase p95 within 10 % of baseline; ADR merged.

Out of scope. The cutover (A9).

A9 · `meta_ads` migration + scraper direct-`pg` cutover · A · L · S2-W2

Read first. A4 artefacts; `services/scraper/lib/db.cjs`; every call site from the A4 inventory; `admin/lib/supabase*.ts`; `admin/lib/auth.ts` (`admin_users`, HS256 cookie) — replaced by Skynet auth for the native pages.

Deliverables. Migration `db/migrations/unified/meta_ads/001_baseline.sql` (schema + all functions, `workspace_id` added to every table with the default workspace, composite uniqueness updated); data migration runbook + script (`infra/migrate/meta_ads/{dump,restore,verify}.sh`, checksums per table); `server.js`, all `lib/*.cjs` *workers and the remaining admin API routes rewired to `lib/db.cjs` (no `supabase-js`, no GoTrue, no Storage client — R2 only, no Realtime); env: `DATABASE_URL` (Skynet Postgres), `R2_`*; the scraper's `settings` table stays, read through the same

client; `node --test test/` extended with a parity suite (old vs new answers for `ads_browse`, `intel_*`, `research_*`, teardown enqueue on the same fixture).

Acceptance. Parity suite green; verify script: row counts + checksums equal after a full migration on staging; `rg "supabase" services/scraper --glob '!node_modules' --glob '!*.md'` = 0 runtime hits.

Out of scope. Native UI (V22/V15); retiring the old stack (A19).

A2d · Scraper engine + transcriber as stack services with hard limits · A · M · S2-W3

Read first. Architecture §5, §9 (re-issued for S2 in ADR-003); `ecosystem.config.cjs`, `Dockerfile`, `transcriber/Dockerfile`; `/opt/meta-ads/docker-compose.yml` on the rescale box (residential proxy settings, `TRANSCRIBER_URL`).

Deliverables. `compose.prod.yml` (+ staging) gains `scraper-engine` (server.js only — no Next admin; `cpus: 6`, `mem_limit: 16g`, `pids_limit: 1024`, `shm_size: 1g`) and `transcriber` (`cpus: 2`, `mem_limit: 8g`, models volume); crons run inside the engine as today; healthchecks; residential proxy + Decodo settings via `.env.production`; A2c's limits table re-issued with these rows; `docs/ops/RESOURCE_LIMITS.md` updated; the old `/opt/meta-ads` stack keeps running until A19 (both write to different DBs — no double-writes: the old one keeps Supabase, the new one Skynet Postgres, and only the new one is scheduled after cutover day).

Acceptance. `docker compose -p skynet ps` shows both healthy; `curl -sf localhost:38000/api/intelligence/healthz`; X3b-style pressure sample within limits.

A13a · Competitors, lenses, schedules — model + migration · A · M · S2-W3

Context. The functional design's model: competitors are workspace entities linked to products, each link carrying market/locale settings; lenses = Ads-Library page, keyword query, domain; schedules per link.

Read first. Functional design §3.0, §3.1; `brands` table + `is_research_lens`; `admin/components/{AddBrandForm,BrandScheduleCard}.tsx`; scheduler cron (`server.js:1516`); D21 rules.

Deliverables. Migration `meta_ads/002_competitors.sql`: `competitors(id, workspace_id, name, domain, notes)`, `competitor_lenses(competitor_id, kind ∈ page|keyword|domain, value, country)`, `product_competitors(product_id, competitor_id, market, locale, auto_recreate_rule jsonb, alert_rule jsonb)`; data migration `brands` → competitors + lenses (1:1, a compatibility view for the engine); the scheduler reads `product_competitors` (interval per link); `pytest apps/api/tests/intelligence/test_competitor_product_mapping.py`.

Acceptance. Test green with one product ↔ three competitors, distinct lenses/schedules; migration verify (brand count = lens count).

Bridge impact. The lander/destination filter proven in bridge I1/I2 becomes lens semantics here (two landers per brand).

Out of scope. Alerts, auto-recreate, APIs (A13b).

A13b · Traction alerts + auto-recreate rules + competitor APIs · A · M · S2-W4

Read first. A13a model; D21 rules; static-ads `competitor_brand_mappings`, `auto_expand_runner.py`; Discord/Telegram senders (X3).

Deliverables. Alert job: traction rule per link (new family ≥ N days running / rank rise) → Discord/Telegram + an outbox event (inbox in S3); auto-recreate rule → Production Expand session via the S1 Expand entry; API `/api/intelligence/competitors*` (CRUD, link/unlink, rules); archive/merge suggestions on competitor links ("silent for N days → archive?", "same family / same library entity as an existing competitor → merge?") as explicit suggestion records, never auto-applied; alert rules create inbox/event artefacts only, never direct actions; `pytest apps/api/tests/intelligence/test_competitor_rules.py`.

Acceptance. Test green; one alert fired and one Expand session created on the fixture.

V22 · Intelligence native pages, group 1 · V · L · S2-W3

Read first. Appendix A.1 rows for brands, browse, families, transcripts, intel, gold, settings, tools; the old pages `/brands*`, `/ads`, `/intel`, `/intel/gold`, `/settings`, `/users`; the A9 API; A13 model.

Deliverables. `apps/web/src/intelligence/`: Competitors list + detail (lenses, schedules, metrics, families), Ads browse (label filters, status, span, collation, family collapse, running-now, video/transcript, sorts, tags, signed media), Intel dashboards (coverage, label distributions, compare), Gold labelling (blind item, label, run evaluation), Settings (models, cost caps, residential, proxies), Tools (manual runs: scrape, crawl, classify, transcribe; backfill family keys; classifier calibration; keyword seeding helpers; probe runs), Rank page (`/rank` + history, freshness diff, rank snapshots), an Intelligence users view that deep-links to Skynet core membership management (login = core auth; the fold is explicit in the checklist). Auth = Skynet roles (`researcher`, `admin`); the old `admin_users` retired with the Next app.

Acceptance. `pnpm --filter web test -- intelligence-group1`; parity checklist `docs/checklists/intelligence-group1.md` with every A.1 row in scope green (screenshots), including `Quick Scrape / rank`, `Settings/users/login` and `Tools`.

A14 · Research native backend + ranked opportunities · A · M · S2-W4

Read first. `lib/research/{runner,cluster,score}.cjs`, `research_*` tables, `research_products_ranked` view, `admin/app/research/*`, `tools/seed-research-keywords.cjs`; D21.

Deliverables. Research runs as Skynet jobs (same engine code over `lib/db.cjs`), keywords/niches per workspace, suggested keywords from tracked-competitor terms and the VOC corpus into a review queue (nothing auto-seeds), runs resumable at boot, ranked products with reasons and gaps, decision tracking on each opportunity (`status`, `reason_code`, `decision_note`, `decided_by`, `decided_at`, `cooldown_until` — default 30 days, hidden from the active view until new evidence); API `/api/intelligence/research/*`; `pytest apps/api/tests/intelligence/test_research_native.py`.

Acceptance. Test green; a staging run over a known keyword set produces the ranked list with reasons.

A15 · Promote to product · A · M · S2-W5

Read first. D22; static-ads product model (`routes/products.py:43,151,248,282`); workos `LISTS.products`, `Product ID`; S8a bridge module (extend with `POST /products` create-or-link).

Deliverables. Action from a ranked opportunity, gated: the researcher recommends, a strategist approves (Mathew for pilot products); on approval create the Skynet product (name, market, brand identity skeleton, docs/looks/offer placeholders flagged `incomplete`, plus the accelerator's client-file fields: *proof we are allowed to use*, *what died / dead angles*, *current winners*), link the competitors and intel from the run, create or link the ClickUp Product card through the bridge (Product ID = naming product number from `rescale_service.next_product_number`), write lineage `research → product`, and create a product readiness checklist (researcher: niche/competitors/evidence; strategist: positioning/claims/brand truth; ops: workflow fields) that must be complete before the product can enter script generation; `pytest apps/api/tests/intelligence/test_promote_to_product.py`.

Acceptance. Test green; one recorded staging promotion (product id, linked competitors, ClickUp card id).

A12a · Teardown native jobs + artefact pages · A · M · S2-W5

Read first. `lib/teardown/{worker,prompts}.cjs`, `ad_teardowns`, `enqueue_brand_teardowns`, `/breakdowns`.

Deliverables. Teardown per family + brand fan-out as Skynet jobs (same worker over `lib/db.cjs`, cost cap + model from settings); the breakdown prompt becomes a versioned prompt asset

(`core.prompt_assets` , Mathew-editable — the first use of the asset model V14 completes in S3); API `/api/intelligence/teardowns*` ; artefact view data for V15; `pytest apps/api/tests/intelligence/test_teardown_jobs.py` .

Acceptance. Test green; one brand fan-out drains on staging; the prompt asset version is recorded on each artefact.

Bridge impact. The Grok `lfs_extract` prompt + schema from bridge I1 is the seed for the teardown prompt asset.

Out of scope. The two actions (A12b).

A12b · Create script / Recreate actions + brief stub / Expand hand-off · A · M · S2-W6

Read first. A12a; functional design user stories 1–2; the S1 Expand entry in `apps/api/production` ; `studio.briefs` stub (until S3).

Deliverables. Create script from this → a script brief entity (`studio.briefs` : teardown ref, evidence packet, product, status) with lineage `teardown → brief` ; **Recreate** → a Production Expand session seeded with the ad's media, lineage `ad → session` ; **Add to brief** → append the teardown/VOC evidence packet to an existing brief; **Tag for teammate** → an inbox item with the selected evidence packet and assignee; when a teardown or research artefact becomes a brief, auto-attach the matching VOC packet (top phrases/threads for the product and selected group, with source metadata and date window), with add/remove controls preserved in the brief payload; `pytest apps/api/tests/intelligence/test_teardown_actions.py` .

Acceptance. Test green; one teardown → one brief + one Expand session on staging (ids attached).

A18 · Reddit VOC pipeline · A · M · S2-W6

Read first. D23; `lib/market-intel/{reddit-client,ingest-worker}.cjs` , `mi_scopes` , `mi_documents` , `mi_document_scopes` , crons `5,20,35,50` + retention `40 3 *` .

Deliverables. `mi_keyword_groups(workspace_id, name, keywords[], scopes[])` (first: cravings, blood sugar); continuous collection per group (scope runs tagged with the group); retention: never purge documents that belong to a group; export `GET /api/intelligence/voc/export?group&from&to` → CSV/JSON (columns per D23); UI in V15.

Acceptance. `node services/scraper/scripts/run-reddit-voc-smoke.mjs --group cravings` collects and the export returns rows; sample attached.

A18b · PubMed evidence adapter + six-lens scopes + MCP + export · A · M · S2-W7

Context. Reddit tells the team how the customer talks about the problem; PubMed tells them why it happens. Six lenses (mechanism, root cause, symptoms, ingredients, failed solutions, desired outcomes) are searched per keyword group and the findings are classified by evidence strength, so briefs and scripts can carry fact-based hooks, authority statements, belief shifts, symptoms to call out, and reasons other solutions fail — in plain customer language, with a citation on every row.

Read first. D23b; `development-processes/pubmed-evidence-layer-plan.md` (§2 lenses + classifier, §3 schema, §4 MCP); `lib/market-intel/{reddit-client,ingest-worker,amazon-worker}.cjs`, `mi_scopes` (`product_name`), `mi_documents`, `mi_document_scopes`; NCBI E-utilities (`esearch` / `efetch` / `elink`, `datetype=edat`, `tool` + `email` required, `NCBI_API_KEY` for 10 req/s).

Deliverables. `lib/market-intel/{pubmed-client,normalize-pubmed,pubmed-classifier,pubmed-worker,pubmed-corpus}.cjs` (plain `fetch` + XML, retry/backoff, rate limit); migration `add_market_intel_pubmed.sql` — `mi_scopes.source` gains `pubmed`, `scope_type` gains `pubmed_query`, `mi_scopes.group_key` (FK to A18's `mi_keyword_groups`) + index, `params = {lens, template_version, terms}`; `mi_documents.doc_type` gains `article`, new `citation TEXT` and `evidence JSONB` (`doi, journal, pub_types[], mesh[], humans, animals, study_design[], sample_n, sample_n_confidence, evidence_class, pmc_id, lens[]`) + index on `evidence_class`; `body` = abstract or title fallback (`metrics.has_abstract=false`), `author_hash` null (public authorship), `purge_after` never set; classifier = publication types + MeSH Humans/Animals + study-design cues + sample-size extractor → classes `strong / moderate / preclinical / case-report / narrative-review / traditional-use / unclassified`; worker watermark on the Entrez indexing date with a 7-day overlap + monthly full re-walk per scope, `watermark_gap` never set; six lens scopes per keyword group seeded from D23b; thin MCP adapter `lib/mcp/pubmed-mcp.cjs` mounted at `/mcp/pubmed` (Streamable HTTP, miAuth token; tools `pubmed_search`, `pubmed_get`, `pubmed_evidence_corpus`, `pubmed_lenses`); `/market-intel/pubmed/{run,stats}`; admin page `admin/app/market-intel/pubmed`; documents route + service types gain the article fields; export `GET /api/intelligence/evidence/export?group&lens&class&from&to` → CSV/JSON (columns per D23b); `test/market-intel-pubmed.test.mjs` with recorded fixtures (a no-abstract record, an animal study, a meta-analysis), the classifier table, sample-size cases, edat overlap, backoff.

Acceptance. Test green; `node services/scraper/scripts/run-pubmed-evidence-smoke.mjs --group cravings` collects all six lenses and the export returns classified rows; an MCP client lists the four tools and `pubmed_search` returns ranked rows on staging; sample attached.

Bridge impact. If R7a is built in the standalone, this card is a port of the proven adapter + MCP into the unified market-intel spine, not a build.

Out of scope. The evidence packet in briefs (A18c); the `/evidence-layer` skill (RescaleOS now, Studio via A11).

V15 · Intelligence native pages, group 2 + tracked-ad actions · V · L · S2-W7

Read first. A12, A14, A15, A18, A18b APIs; old pages `/research*`, `/breakdowns`, `/market-intel/*`.

Deliverables. Research (keywords, runs, ranked products with reasons, Promote to product), Breakdowns (list, detail, brand fan-out), Market intel (scopes, documents, threads, aliases, stats; VOC groups + export; evidence view with `pubmed` source filter, lens + evidence-class filters, export), tracked-ad actions on any ad/family: Expand (Production), Create script, Recreate; static-ads' scraped-ads pages retired (collections become tags on the one corpus).

Acceptance. `pnpm --filter web test -- intelligence-group2`; staging walkthrough research → product → tracked ad → Expand / script (recording).

A19 · Scraper retirement + duplicate-scraper shutdown + archive · A · M · S2-W8

Read first. A2d (both stacks), `/opt/meta-ads` compose, static-ads `scraped_ads.py`, `decodo_client.py`, `scrape_runner.py`, `ad_collections`, bridge cards I1/I2/R7a.

Deliverables. Cutover day: final `meta_ads` delta migration (verify script), old crons stopped, `meta-ads.rescale.media` → redirect to `app.rescale.media/intelligence`, old Supabase project read-only then archived (dump to the Storage Box), old container removed from `/opt/meta-ads`; static-ads' scraper routes removed from `apps/api`, collections migrated to tags; bridge export surfaces retired (`/bridge/lfs/runs*`, the package route, the Primal Queen continuous folder-drop path, any bridge-only schedule/worker, `is_long_copy_native`, the `lfs_extract` tier after A12a absorbs it); `docs/runbooks/S2-intelligence-cutover.md`.

Acceptance. Checklist: old cron off, redirect live, `rg "bridge/lfs|lfs_extract|BRIDGE_DROP_ROOT" services/scraper` = 0, `rg decodo apps/api` = 0, archive dump present; X3 checks updated.

Bridge impact. Retires every scraper-side bridge path proved in I1/I2/R7a.

Out of scope. The console (A20).

A18c · Evidence packet in briefs · A · S · S2-W8

Read first. A12b (VOC packet auto-attach), A18b (`pubmed_evidence_corpus`), `studio.briefs` stub.

Deliverables. When a teardown or research artefact becomes a brief, the evidence packet (top findings per lens for the product and group: plain-language finding, citation, evidence class, PMID link) is auto-attached next to the VOC packet, with add/remove controls preserved in the brief payload; the packet carries the claims-policy flags (`form-mismatch`, class) so script generation and the compliance pass

read them from the row, never from the prose; `pytest apps/api/tests/intelligence/test_evidence_packet.py`.

Acceptance. Test green; one staging brief shows both packets (ids attached).

A23 · Chat source adapters: Telegram + Discord channel ingestion · A · M · S2–W7

Context. Selected Telegram/Discord channels become sources in the market-intel corpus (customer/community channels) and later feed the feedback lanes (team channels, S3). Consent per channel; authors salted-hashed like every other market-intel source.

Read first. `lib/market-intel/{ingest-worker,reddit-client}.cjs`, `mi_scopes` (`scope_type`), `mi_documents`, `MARKET_INTEL_AUTHOR_SALT`; RescaleOS `console/src/server/cron/feedbackPoller.ts`, `console/src/server/feedbackPollers/{telegram,discord,secrets}.ts` (bot polling, offsets, attachment capture); Discord bot install pattern in ecomprofits `notification-platform/src/server/discord-install.service.ts`.

Deliverables. `mi_scopes.scope_type` gains `telegram_channel` and `discord_channel` (bot token per workspace from `core.integration_credentials`, channel id, consent flag, keyword group); a poller worker (long-poll Telegram `getUpdates` with offsets; Discord gateway or channel REST with `after`) writing `mi_documents` (+ thread reconstruction like Reddit); retention per D51; searchable in the VOC UI (V15) by source `telegram|discord`; `node --test services/scraper/test/chat-adapters.test.js`.

Acceptance. Test green with recorded fixtures; one consented channel ingests on staging and appears in the VOC search with the author hashed.

Out of scope. Team-feedback lanes and the agent (S3, A24); DMs; channels without consent.

S3 — Studio merge + strategy layer (specs)

Facts every S3 card relies on (verified against `RescaleOS`): console = Next.js (`console/`), SQLite `console.db` in `DATA_DIR` (`console/src/db/client.ts`, ~29 tables: `sessions, session_events, gates, artifacts, cron_runs, settings, claude_account_state, claude_account_windows, product_settings, research_ads, users, auth_sessions, scripts, script_archive, script_revisions, script_votes, weekly_results, feedback_notes, feedback_inbox, feedback_lanes, fathom_links, notifications, audit_log, backlog_items, backlog_digests, backlog_ideas, supermemory_writes, playbooks, session_sources`); agent = `@anthropic-ai/claude-agent-sdk` spawning the Claude Code binary with a subscription OAuth token (`managedSession.ts`, `sessionManager.ts`, `accounts.ts` rotation); engine API `/api/engine/` (*`x-cron-token`*), *research ingest* `/api/research/ingest`, *assign* `/api/assign` (*`adName.ts`*), *results* `/api/scripts/results`; *loops:* `engine/automation/` (`npt-build-loop.sh` hourly, `frankenstein.sh`, `dream-daily.sh`, `backlog-daily.sh`, loop-closer, trend-watcher, volume-

trigger, lint, backup), cloud runner `console/scripts/night-shift-runner.mjs` + `night-shift-jobs.cloud.json`, registry `console/src/server/cron/registry*.ts` (`RESCALEOS_CLOUD=1`); skills: `engine/skills/` = the versioned mirror of the live `~/.claude/skills` (45 skill dirs, each `SKILL.md` + assets; `engine/skills/_doctrine/` 13 shared doctrine files; `engine/memory/` 41 wikilinked corpus pages), re-synced by `/backup`; the cloud image replaces `~/.claude/skills` on boot; **there is no `workspaces/` directory** — business workspaces are repo-root dirs `dropshipping/`, `supplements/`, `strategic-loop/` (each with `CONTEXT.md`, automations, product folders); engine API = 18 routes under `console/src/app/api/engine/*` (`backlog/{accepted,applied,digest}`, `fathom`, `fathom/processed`, `feedback`, `inbox`, `kpi-reminders`, `loop-close`, `playbook`, `playbook-stats`, `query`, `scripts`, `scripts/content-patch`, `session-sources`, `supermemory/winners`, `uploads/[...path]`, `upsert`), contract in `_doctrine/console-scripts-api.md`; console pages: dashboard, login, sessions, sessions/[id], scripts, scripts/[ws]/[id], backlog, backlog/[date], inbox, night-shift, scoreboard, opportunities, audit, users, admin/claude-usage, admin/supermemory; SDK spawn (`managedSession.ts`): `query({ options: { model, cwd: WORKSPACE_ROOT, settingSources: ['user','project'], permissionMode: 'acceptEdits', allowedTools: [Read, Grep, Glob, Skill, Task, TodoWrite, WebFetch, WebSearch, Write, Edit], mcpServers: {trendtrack?}, maxTurns: 400, env: {CLAUDE_CODE_OAUTH_TOKEN: account.token}, resume?, canUseTool } })` — no `systemPrompt`; behaviour comes from the skills/settings sources + the workstream slash command as the first message (`sessionManager.ts:216–221`, AUTOPILOT policy); night-shift cloud jobs (`night-shift-jobs.cloud.json`, UTC): `npt-loop-closer` 03:11 daily, `frankenstein` Mon 06:13, `npt-trend-watcher` Mon 07:23, `volume-trigger` Wed+Fri 07:47, `backlog-feedback` 05:03 daily; Mac registry adds `npt-build-loop` (3600 s), `dream` 04:31, `backlog` 05:03, `lint` Sun 05:17, `backup` 23:53; SuperMemory `winnerHook.ts`; workstreams `npt | bpt | npt-niels` (`src/lib/workstreams.ts`).

A10 · Agent runtime on Postgres `studio` + engine API parity · A · L · S3-W1

Read first. D31; `console/src/db/client.ts` (DDL) + `dal.ts` ("the Postgres seam"); every `/api/engine/` *and* *`/api/research/`* route; `console/scripts/night-shift-runner.mjs`.

Deliverables. Migration `db/migrations/unified/studio/001_baseline.sql` (every SQLite table → Postgres with `workspace_id`, JSON columns as `jsonb`, `session_events` append-only); DAL rewritten on `pg` (same function signatures); data migration script SQLite → Postgres with row-count verify; engine API unchanged in shape (`x-cron-token` replaced by the Skynet service token, one adapter); the console runs as service `studio` in the stack (Node 24, `cpus: 4`, `mem_limit: 8g`) reading the new DB; `pnpm --filter agent-runtime test` + `test_studio_pg` parity suite (every DAL function on both backends over the same fixture).

Acceptance. Parity suite green; migrated data verified; a session starts, streams, gates and completes on Postgres.

A11 · Runtime assets, night-shift on the stack, workspaces, subscription binding + rotation · A · L · S3-W2

Read first. D31; `engine/` layout (skills, doctrine, memory, playbooks), the repo-root business workspace dirs `dropshipping/`, `supplements/`, `strategic-loop/`, `engine/automation/*`, `night-shift-jobs.cloud.json`, `registry.mac.ts`; `accounts.ts` (`CLAUDE_ACCOUNT_{i}_NAME/TOKEN`), `/admin/claude-usage`.

Deliverables. Skills, doctrine, playbooks and the business workspace dirs (`dropshipping/`, `supplements/`, `strategic-loop/`) become repo-tracked runtime assets mounted read-only into the `studio` service (no laptop copies; `/evidence-layer` + `_doctrine/evidence-contract.md` included, its MCP transport bound to A18b's `/mcp/pubmed`); every loop (`npt-build-loop`, `dream`, `lint`, `backlog`, `loop-closer`, `frankenstein`, `trend-watcher`, `volume-trigger`, `backup`) registered in one scheduler on the stack (`studio.cron_jobs`, UTC schedules recorded, per-job limits); the Claude subscription runner = binding `studio.agent` in `core.model_bindings` with the account rotation intact and usage posted to `usage_events`; `/admin/claude-usage` data in the Skynet admin; `node services/agent-runtime/scripts/run-night-shift-smoke.mjs`.

Acceptance. Smoke green (each job runs once on staging); `curl -sf /api/admin/claude-usage` shows rotated accounts; Bian's Mac launchd jobs disabled per D31's window.

A21 · Studio parity APIs · A · L · S3-W3

Deliverables. Every console-facing API the native pages need, on the Postgres DAL: scripts (list/detail/revisions/archive), sessions (start/stream/gates/artifacts), scoreboard/weekly results, opportunities + votes, backlog (digests/ideas/accept), editors/assignment (`/api/assign` semantics; the RescaleOS ad-name format retired in favour of the creative code + v3 name from S3/S8a), notifications, playbook, research ingest, KPI reminder (now a readback check), feedback pollers (Telegram/Discord) state; route map old → new in `docs/checklists/studio-api-parity.md`.

Acceptance. `pnpm --filter agent-runtime test -- studio-parity` green; the parity map complete.

V14 · Prompt layers + per-user overlays + Mathew-editable assets · V · L · S3-W3

Read first. Functional design §3.0 (layers: doctrine → workflow → brand/product → personal overlay), TrenchOS digest (authoring guardrails), D32; A12's `core.prompt_assets` stub.

Deliverables. `core.prompt_assets(id, workspace_id, layer ∈ doctrine|workflow|brand|user, key, version, body, author_id, status)` + resolver (`resolve_stack(workflow_key, product, user)` → ordered layers, recorded on every run as `resolved_stack`); editor UI with versions/diff/publish, negation lint and positive-only checks, **and an eval step before publish** (a

benchmark set per layer — 5–20 fixed inputs with expected properties, run against the new version and the current one, results side by side; the pattern of Anthropic's `skill-creator`), role-gated (Mathew/admin for doctrine + workflow; users for their overlay); static-ads `instructions` migrated as the first user overlays; `evidence-contract` (evidence classes + claims policy) seeded as a doctrine-layer asset.

Acceptance. `pnpm --filter web test -- studio-prompt-layers`; one recorded edit → re-run shows the new version in `resolved_stack`.

Bridge impact. Seeds this layer system with the Grok extraction schema, the "write one for me" packet contract, and `rules.md` from bridge I1/I3/I4.

S16 · Script-entity bridge to ClickUp Script cards + demand-line hand-off · S · L · S3-W3

Read first. D33; workos `forward.ts:186` (`handleScriptApproved`), `LISTS.scripts`, `demand-week.ts`, `spawn.ts`, `qa-gates/*`; S8a bridge module.

Deliverables. Bridge endpoints in workos: `POST /scripts` (create the ClickUp Script card from a Skynet script entity, structural fields only per invariant 8c), `PATCH /scripts/:id` (human-owned fields), webhook → Skynet on Script status changes (approved → Creative spawn continues in workos); `POST /demand-lines` (from allocation, S3-W6) using the spawn path; Skynet keeps `studio.scripts.clickup_task_id`; reconciliation report both ways; `vitest services/workos/tests/script-bridge.spec.ts`.

Acceptance. Test green; bidirectional transcript for one script (Skynet → card → approved in ClickUp → Skynet sees it).

V23 · Script Studio native pages · V · L · S3-W4

Deliverables. `apps/web/src/studio/`: scripts list + detail (creative DNA, revisions, votes, assignment, sign-off, learnings, handoff), sessions (live stream, gates: reply/question/permission, artifacts, save handoff), scoreboard (readback-fed), opportunities, backlog, overview, night-shift, admin usage, audit, users (deep-linked to core membership), admin/supermemory; a compact handoff artefact (current state, decisions, changed artefacts, open loops, the exact next step, a resume prompt) that can seed the next run; old console URL → redirect after A20.

Acceptance. `pnpm --filter web test -- studio-shell-pages`; walkthrough recording (scripts → session → gate → sign-off).

S14 · Inbox + dispatch · S · M · S3-W4

Read first. `core.events_outbox` (S4); functional design §3.2, user stories 9, 15, 27.

Deliverables. `core.inbox_items(workspace_id, owner_id, kind, source_ref, evidence jsonb, state ∈ open|done|dismissed, created_at)` ; `core.chat_feedback_captures(id, workspace_id, source, external_id, channel_id, author_hash, text, attachment_ref, posted_at, ingested_at, processed_at)` ; port the RescaleOS feedback pollers into Skynet for registered team channels (Telegram `getUpdates` offset, Discord REST `GET /channels/{id}/messages?after=` with attachment download) so team-feedback channels feed the inbox/dispatch flow here, while customer/community channels remain A23 market-intel sources; producers: script comments/tags, teardown tags, hook kills, chat-feedback captures, alerts (A13b), bridge actions, and manual rough capture (drop a file, link or half-thought into the inbox unsorted, then process or file it later); dispatcher rules (who gets what); UI inbox in the shell with quick actions ("assign to X", "start a script", "process now", "file later"); Telegram/Discord quick replies land here (delivered by S27, acted on by A24); `pytest apps/api/tests/studio/test_inbox_dispatch.py` .

Acceptance. Test green; one staging item with owner, evidence packet, resolution.

V16 · Hook session + approval gates · V · L · S3–W5

Read first. Functional design §3.4 (hook gate), Hook Engine digest (seed content, claim validation, brand-tone band), D32; V14 resolver; readback hook rate (W12).

Deliverables. Hook session flow: over-produce (N hooks per brief) → cull (brand rules, claim validation, stop-scroll judge) → survivors → strategist picks keepers / kills with reasons (recorded as signals for V17); gates per asset or batch per D32, with owners and evidence; hook rate shown from readback per launched hook; `studio.hook_sessions` , `studio.hook_candidates` .

Acceptance. `pnpm --filter web test -- hook-session` ; a saved session artefact with generated / survivors / kill reasons.

S15 · Lineage producers (scraper, studio, workos) + coverage store v0 · S · M · S3–W5

Deliverables. Lineage nodes/edges from: Intelligence (ad/family → teardown → brief), Studio (brief → script → session → gate), workos bridge (script card, creative card, launch); `core.coverage_facts(workspace_id, product_id, segment, awareness, angle, source ∈ label|creative_code|performance, count, evidence_refs[])` filled from labels (A9 data), creative codes and snapshots; validated-memory writeback: `core.validated_learnings(workspace_id, product_id, kind ∈ winner|dead_angle|rule, statement, rule_taught, evidence_refs[], source_event, confirmed_by, created_at)` written only from human-confirmed winners **and confirmed losers** (S9 — a dead angle records the rule it taught, first-class like a winner) or human-rated outputs; promotion threshold per the accelerator data dictionary (second occurrence = pattern proposal, third = rule/workflow proposal) — personal taste signals from V17/V19 stay in the instinct stores and never write here; the SuperMemory writer moves behind this table; `pytest apps/api/tests/core/test_lineage_producers.py` .

Acceptance. Test green; one chain teardown → script → export → performance → winner candidate; one confirmed winner writes one validated-learning record; one confirmed loser writes one `dead_angle` record with `rule_taught`; one unconfirmed edit writes none.

S5b · Winner/loser rule evaluation · S · M · S3-W6

Read first. workos `platform-sync/classify.ts` (zombie ≤ BEROAS×1.115; winner ≥ target ROAS 15 % and spend ≥ 5× target CPP; super-winner + 7-d spend ≥ €1 000; TOF review zone), `thresholds.ts` (Google Sheet knobs); ecomprofits `business-logic.ts`; `core.performance_snapshots`.

Deliverables. `core.classify_rules(workspace_id, product_id?, knobs jsonb, version)` (knobs migrated from the sheet, editable in admin); evaluator over snapshots → `core.winner_proposals(creative_code, verdict, evidence, rule_version)`; fixture parity vs workos thresholds.

Acceptance. `pytest apps/api/tests/studio/test_winner_rules.py` green with the parity fixture.

S9 · Winner/loser human gate + winner lineage · S · M · S3-W6

Deliverables. Proposals (winner **and loser**) surface in the inbox (S14) and the winners view; confirm/override with reason → `winner` or `dead_angle` lineage node + event, the loser carrying the rule it taught (S15); resolved status exposed to Studio (scoreboard) and Production (winners gallery); the workos platform-sync flag stays as a mirror until S4.

Acceptance. `pytest apps/api/tests/studio/test_winner_gate.py` green; one confirmed winner emits the lineage event; one confirmed loser emits a `dead_angle` lineage event with reason + `rule_taught`; the API shows both.

V18 · Coverage map + gap finder + allocation + brief queue · V · L · S3-W6

Read first. Functional design §3.3 (Mathew's system map), D32; S15 coverage facts.

Deliverables. Coverage map per product (segments × awareness × angles) with the evidence-count honesty guard (cells show counts + confidence, never a bare heat colour); gap finder ranked by importance × reach; allocation view where the team lead sets net-new / variations / scaling (human gate, pre-filled from last batch's results and the gap ranking) → brief queue (`studio.briefs`) — each brief packet carries **swipes from a curated swipe library** (`studio.swipes(id, workspace_id, mechanism, source_ref, media_ref, one_thing_to_steal, added_by, visibility ∈ personal|shared, approved_by, shared_at)`, organised by mechanism, not by brand — the accelerator convention; personal swipes stay personal until explicitly promoted to `shared`) next to its teardown / winner / market-intel / coverage evidence → Script Studio; each brief packet carries

market-intel corpus evidence (customer phrases), teardown evidence, winner evidence and coverage evidence, co-selectable, and the review screens show those packets explicitly; rejected proposals persist with reason codes; weekly proposal batch generator (review, not auto-run).

Acceptance. `pnpm --filter web test -- coverage-map` ; one reviewed allocation batch linked to brief cards.

V17 · Instinct loop for hooks · V · M · S3-W7

Read first. Instinct Transfer digest, functional design §3.4 (instinct loop rules); V16 signals.

Deliverables. Per strategist × deliverable type: exemplar pool (shipped or human-rated only), two-tier rules (hard → auto-reject; tendency with context), before/after pairs, a judge scoring drafts before review; captures are proposed from edits/picks/kills and ratified by the user; repeated signals surface as proposals (second occurrence = pattern/tendency proposal; third occurrence = reusable rule/workflow proposal); two explicit ritual moments in the UI — end-of-session **reflect** ("what is worth keeping, and where: rule / tendency / exemplar / dead angle") and mid-session **remember this** (a standing rule stated in chat or review becomes a proposed capture) — per the accelerator skills; rules apply to the author's drafts after their review, team-wide only after approval; rule audit view.

Acceptance. `pnpm --filter web test -- instinct-hooks` ; one ratified rule capture attached to a hook review.

Bridge impact. Miguel's reaction-banking rule from bridge I4 is seed content for the instinct captures and ritual moments here.

V19 · Instinct loop for scripts + judge + hypothesis ledger · V · M · S3-W7

Deliverables. Same loop for script drafts, with the same reflect / remember-this ritual moments; the judge on drafts; `studio.hypotheses(brief_id, statement, expected_signal, outcome, evidence)` ledger linked to briefs, sessions and results; the third occurrence of a repeated move proposes a reusable workflow/skill (save-as-skill, with approval).

Acceptance. `pnpm --filter web test -- instinct-scripts` ; one draft shows judge output + a linked hypothesis.

V20 · Layer-authoring assistants, template sync, video recipes · V · M · S3-W8

Read first. TrenchOS digest (layer-authoring assistants, template fork + upstream sync, provider adapters as data), functional design §3.5 (recipes).

Deliverables. Assistants that help write a layer with the guardrails (negation lint, context-block authority); templates with fork + upstream sync, each template/skill change run through the V14 eval step before publish; `production.recipes` registry (statics + video) over code-backed executors, statics-first graduation to video recipes.

Acceptance. `pnpm --filter web test -- studio-assistants`; one synced template + one video recipe on staging.

A20 · Standalone console retirement + archive · A · M · S3-W9

Deliverables. Per D31: Mac launchd jobs off, console URL → redirect to `/studio`, SQLite archived (WAL-safe copy to the Storage Box), `/opt/rescaleos` stack removed, bridge-only skills and transport notes retired (`/lfs`, `/evidence-layer` bridge transport wording where the Studio-native flow replaces it, R7 MCP wiring), `docs/runbooks/S3-studio-cutover.md`; comms to Bian.

Acceptance. Checklist: authoring jobs run only on the stack (job log), old URL redirected, archive present, `rg "/lfs|evidence-layer" services/agent-runtime apps/web/src/studio docs` shows only native/live references.

Bridge impact. Retires every console-side bridge path proved in MC1/I3/R7.

Out of scope. Anything the Studio-native flows have not yet replaced (stays until they do).

S27 · Notification platform port (registry, outbox, dispatcher, destinations, mutes, digests) · S · L · S3-W7

Context. Every job and rule in Skynet needs one outbound path with dedup, routing and mutes — ecomprofits already has it; Skynet ports the pattern (Python) rather than re-inventing it.

Read first. ecomprofits `packages/features/notification-platform/src/server/{emit,dedup,destinations,mutes,dispatch,deliver,api,cutover}.ts`, `events/registry`, `schemas/index.ts` (severity, EmitEventOptions), `naming-conventions/src/server/naming-violation-event-factory.ts` (an event family with filters + attachment); workos Discord/Telegram senders; X3's alert senders (to be folded in).

Deliverables. `core.notification_events(id, workspace_id, feature, event_type, severity, payload jsonb, dedup_key, final_status, created_at, dispatched_at)` (unique `(workspace_id, dedup_key)`); `core.notification_deliveries(id, event_id, workspace_id, provider, channel_type, destination_fingerprint, destination_metadata jsonb, status ∈ pending|processing|sent|failed|dead_letter|muted|skipped|retry_scheduled, attempts int default 0, next_attempt_at, locked_at, locked_by, provider_message_id, provider_response jsonb, last_error, created_at, updated_at)`; `core.notification_destinations(workspace_id, channel_type ∈ discord|telegram|email|webhook, purpose, config_encrypted, is_enabled)`; `core.notification_mutes`; event registry ( `apps/api/core/notify/registry.py`, typed payloads per feature: intelligence, studio, production, workspace, admin, spend_guard, synthetic); `emit()` →

outbox; dispatcher job + sweeper with delivery claim / retry / finalize helpers (idempotent on `provider_message_id`); digests (daily ops digest); routing is purpose-based (no subscription UI yet): one enabled destination per `(workspace_id, purpose, channel_type)` with deterministic fallback rules in `docs/contracts/notifications-routing.md`; message renderers per channel with quick-action buttons (Discord components / Telegram inline keyboards) that call Skynet actions through the same permission checks (A24 wires the handlers); initial senders migrated onto `emit()`: X3, A13b, S9, V4 and every new S3 job; `pytest apps/api/tests/core/test_notifications.py`.

Acceptance. Test green (dedup, mute, routing by purpose, digest); on staging one event of each feature reaches Discord and Telegram; no inline `requests.post` to a chat API remains (`rg` = 0).

Out of scope. Chat ingestion (A23); the agent (A24).

A24 · Agents in chat: bot proposals with quick actions, @skynet tag-and-ask · A · L · S3-W8

Context. The bot tags humans with proposals when work finishes ("scrape done — 3 new families, 1 alerted; teardown / add to brief / ignore"), and the team can tag the bot in a thread to ask or act.

Read first. S27 renderers + quick-action contract; RescaleOS `managedSession.ts` (session with context), `feedbackPollers/*`; S14 inbox; the permission catalogue (S1a).

Deliverables. Quick-action handlers (Discord interaction endpoint, Telegram callback queries) → Skynet actions (teardown, add to brief, promote, confirm winner, assign) with the actor resolved to a Skynet user (chat identity ↔ user mapping, `core.chat_identities`) and permission-checked; `@skynet` / bot mention in a thread → an agent session with the thread + detected entities (creative code, product, ad link) as context, answers in-thread, may act through the same handlers; transcript + outcome attached to the inbox item; rate limits and a kill switch; `pytest apps/api/tests/core/test_chat_agent.py` (mocked chat APIs).

Acceptance. Test green; staging demo: a finished research run posts a proposal with buttons, a human presses "promote" and the product is created with lineage; an `@skynet` question about a creative code answers with its hook rate.

Out of scope. Voice; DMs; channels not registered as destinations.

S4 — Workspace cutover (specs)

Facts every S4 card relies on (verified against `rescale-workos`): 23 ClickUp lists in `service/src/domain/ids.ts` (team `90121818343`); the 13 creative-workflow lists in scope: `researchIntake, products, competitors, scripts, creatives, funnelsLanders, ugcCreators, launches, emailFlows, creativeDemand, campaigns, adsets, ripSources`; cascade modules `demand-week.ts` (paste → parsed → locked → spawn), `spawn.ts`, `forward.ts` (`handleScriptApproved`, `handleCreativeApproved` → Launch), `apply-names.ts`, `reconcile-cron.ts` (15-min sweeps), `platform-sync/` *(15-min readback)*, `qa-gates/`, `bounty*` (weekly payroll — stays), `launch-orphans.ts`; ClickUp client `clients/clickup.ts` (700 ms rate guard, `resolveField` by name); Railway project "Rescale Tools", `railway.json`, `nixpacks.toml`;

`rescale_service` schema in ecomprofits Postgres (sequences, `get_creatives_ranking`); field sets per list in `FIELDS.<list>` (`auto: true` = machine-written): researchIntake (Signal strength, Signal note | Legacy ID), products (24 human incl. Brand, Market, Supplier link, Niche, Evidence, Channels, offers… | auto Product ID, COGS, Margin %, Breakeven ROAS, Target ROAS 15/20 %, Target CPP, Shopify SKU, RESEARCH-ID, Currency…), competitors (Active ads, Total ads, Language, Competitor type, Ad library link), scripts (Hook type, Hooks, Angle, CR_AdCopy, CR_Headline, Script type, Proof needed, Approach, CR_avatar, CR_Hypothesis, Loom link, Script URL | auto Sequence #, Week/Batch #, Is image?, Image sub-type), creatives (Format, CR_fileurl, Ad copy, Headline, Script URL, CR_Hypothesis, CR_avatar, CR_tool | auto Market, Week/Batch #, Spawn #, CR_ex…CR_editingrhythm, Rip source code/URL, Source winner codes, Creative code), funnelsLanders (Funnel type/Flow, Flow code, Angle, Price point/visibility, Fast shipping, Payment, Channel | auto Funnel URL, version #, task kind, Funnelish name/ID, Domain, Available?), launches (Channel, Market, Headline, Ad copy, Error, Error reason, Objective | auto Ad name NC, Creative URL, Platform ad ID, Spend to date €, ROAS, KPI met, Classification note, Launched date, Zombie, Ad account, Credit line, Pixel, Campaign name NC, Adset name, Funnel missing?), emailFlows (Flow type, Channel, Creator/Agency | auto Klaviyo flow name), creativeDemand (Raw post, Notes, CR_Hypothesis, CR_avatar | 20 auto incl. Week label, Totals, Spawned?, Produced target/count, Fulfilled?), campaigns (Type, Platform, Objective | auto CBO budget, Market, Increment #, Campaign name, Platform campaign ID, Spend €, ROAS), adsets (all auto), ripSources (Source ad link, Market, Channel, Creative type, CR_avatar | auto Assigned?, marker); **`ugcCreators` (list 901218779868) has no field definitions at all** — D42 must define it before S19; status templates (`scripts/field-setup/fields-manifest.json`): `production_lifecycle` Briefed → In Progress → In Review → Needs Edits → Approved → Live / Learning → □ Winner → □ Super Winner → Loser; `product_pipeline` Candidate → Vetting → Testing → □ Winner → Retired → Rejected; `simple_flow`; per-list overrides for Products, Scripts (adds □ Idea, Idea approved, Proof), Creative Demand (Draft, Parsed, Locked, Delivered), Campaigns (Draft, Live, Exhausted, Closed), Adsets (Active, Archived).

S0b · Pre-pilot discovery (per-brand launch process) · S · M · S4-WO · `docs/s/s0b-launch-discovery`

Context. The launch board is the *proposed* first native board. Gate G2 confirms or re-targets it. Without a real map of how each brand launches statics today, S8's data model is a guess.

Read first. Functional design §3.6 (launch with a creative code) and §7 item 10; Native Workspace PRD (`rescale-workos-native-workspace-prd.md`) phases A–D; workos `service/src/domain/ids.ts` lines 187–210 (the 23 Launch fields) and `service/src/modules/cascade/launch-outcome.ts` (outcome statuses); `rescale-workos-handoff.md`.

Deliverables. `docs/discovery/launch-process.md`: (1) one table per brand with columns request source · approval chain · asset packaging (Drive/ClickUp/other) · naming pattern used · account destination · who publishes · time from request to live · where it breaks (with an example); (2) a list of the fields a media buyer actually reads before launching (compare with the 23 ClickUp fields — which are dead); (3) the proposed pilot product + media buyer, with why; (4) two interview notes (media buying, ops), 45 min each.

Interfaces. Feeds S6 (G2 write-up) and S8 (data model must cite the table).

Steps. Book both interviews; run them plus one screen-share of a real launch; draft; then S6.

Acceptance. Doc merged; every active launch brand in scope has a row; pilot product and media buyer named; Agon and Vince have read it (comment on the task).

Out of scope. Designing the board UI (V9); ClickUp changes.

S6 · Discovery write-up → Gate G2 · S · S · S4-W1

Context. Decides whether the launch board is the first native board.

Read first. S0b doc; lock doc §3 (G2 exit + fail path); technical plan §4.1.

Deliverables. `docs/decisions/G2-workspace-board-order.md` : decision (launch board first — go / re-target / defer), pilot product, media buyer, the S8b field list derived from the discovery table (which of the 23 Launch fields the app shows/writes), and the ordered list of the remaining 12 creative-workflow boards from D42. Gate review with Agon (+ Vince).

Acceptance. ADR merged before S8b starts; S8b's PR cites it.

Out of scope. Anything after G2.

S8b · Launch board API + ClickUp mirror · S · L · S4-W2

Read first. S8a module (extend); lock §4.5 (status classes, field ownership, per-field compare-and-write, durable URLs); `forward.ts:550` , `LISTS.launches` , `FIELDS.launches` , `reconcile-cron.ts` , `launch-outcome.ts` , `clients/clickup.ts` ; D41, D42.

Deliverables. Full launch API in workos (`GET /launches` , `GET /launches/:id` , `POST /launches` with Creative + Product relationships, `PATCH /launches/:id` per-field compare-and-write with `conflicts[]` , outcome statuses observe-only → 409), idempotency table `rescale_service.launch_writes` , webhook → Skynet on every launch change, native store `workspace.launches` in Skynet as a **mirror** (ClickUp still authoritative until the board flips), reconciliation job both ways with a report; `LAUNCH_LIST_ID` sandbox; `vitest services/workos/tests/launch-board-mirror.spec.ts` .

Acceptance. Test green; bidirectional create/update transcript; 24-h reconciliation report on staging = 0 diffs.

V9 · Launch board native UI · V · L · S4-W2

Deliverables. `apps/web/src/workspace/launch-board/` : queue, detail (package, code, ad name copy, assets), status transitions (pre-launch only), assignee, filters, conflict banner, reconciliation state; flag

`launch_board` per workspace.

Acceptance. `pnpm --filter web test -- launch-board`; the named media buyer completes one launch end-to-end on staging (recording).

S11 · Pilot SOP + walkthrough → G3 · S · S · S4–W3

Context. The launch-board pilot may only move to production dual run after the buyer workflow, reconciliation behaviour and rollback path are proven on one named product.

Read first. S6 (`docs/decisions/G2-workspace-board-order.md`), S8b API + mirror contract, V9 UI, S10 loop E2E runbook, X5 rollback drill, S12 migration rehearsal, D41.

Deliverables. `docs/pilot/launch-board-sop.md` (who does what, when ClickUp still rules, how conflicts are resolved, proof-of-launch loop); recorded walkthrough with the named media buyer on staging; pilot-product data-hygiene checklist (assets, naming, assignees, package fields, mirror health); final reconciliation rerun over the pilot window with a diff report (0 unexplained diffs); `docs/decisions/ADR-004-pilot-go-live.md` with go / no-go and the exact feature-flag state for production.

Acceptance. SOP merged; walkthrough recording attached; reconciliation report with 0 unexplained diffs; ADR-004 merged before S4-W4 starts.

Out of scope. Building the board; ClickUp schema changes; boards beyond Launches.

S26 · Board batch 2 API + mirrors: creativeDemand, scripts, creatives · S · M · S4–W4

Read first. D42 (authority, ownership per list), `demand-week.ts`, `spawn.ts`, `forward.ts`, `qa-gates/*`, `fulfillment.ts`; S16 (script bridge — becomes the native store).

Deliverables. Native stores `workspace.demand_weeks/demand_lines`, `workspace.scripts` (merging `studio.scripts` link), `workspace.creatives`; the cascade logic (parse → lock → spawn → approve → creative → launch) runs in workos against the native stores with ClickUp as the mirror (write-through both ways) until the board flips per list; QA gates ported (`M4`); the Creative analyzer (`M5`: video/audio evidence inputs, the 17 CR_* fields auto-filled on Creative entities) ported onto the native Script/Creative approval surfaces; dual-run reconciliation report; `vitest services/workos/tests/board-batch2.spec.ts`.

Acceptance. Test green; one dual-run report (both directions, 0 unexplained diffs); owner sign-off per list.

V24 · Board batch 2 native UI · V · M · S4-W4

Context. Batch 2 moves the highest-coupling creative-workflow boards native while ClickUp remains the rollback mirror.

Read first. S26 API contract, D42 board-authority doc, S16 script-bridge state map.

Deliverables. Boards for Demand (weeks, paste → parsed lines, bounce rows, lock), Scripts, Creatives (structural fields, briefs, QA-gate status, assignment, spawn visibility), rollback state per board.

Acceptance. `pnpm --filter web test -- workspace-batch2`; parity checklist signed by the list owners.

Out of scope. Launches, Products, Competitors, Research Intake, Rip Sources, Campaigns, Adsets, Email Flows, company-ops lists.

S18 · Board batch 3 API + mirrors: products, competitors, researchIntake, ripSources · S · L · S4-W5

Context. Batch 3 moves the entity/intake boards native so product setup, competitor tracking, intake and rip sourcing no longer depend on ClickUp as the active workflow store.

Read first. D42, A13a/A13b, A15, the workos reconcile rules (`reconcile-cron.ts`, provenance sweeps), the economics sync modules (`modules/economics/*`, `thresholds.ts`).

Deliverables. Native stores + mirrors for the four lists; products link to the Skynet product (A15) and the naming product number; competitors link to A13 entities; research intake feeds A14; rip sources feed Expand; provenance/reconcile sweeps ported; `M8` Sheet sync + unit economics parity (COGS, margin %, breakeven ROAS, target ROAS 15/20 %, target CPP, currency — the sync/reconcile rules that fill those fields today, sourced from the economics modules) on Products; `vitest services/workos/tests/board-batch3.spec.ts`.

Acceptance. Test green; dual-run report; owner sign-off.

Out of scope. Demand, Scripts, Creatives, Launches, the final read-only cutoff, company-ops lists.

V25 · Board batch 3 native UI · V · L · S4-W5

Context. Batch 3 moves the entity and intake boards native so research/product setup no longer depends on ClickUp for creative-workflow work.

Read first. S18 API + mirror contracts; A13 competitor entities; A15 Promote to product; D42 board-authority doc.

Deliverables. Native boards for Products, Competitors, Research Intake and Rip Sources with per-board rollback state, owner/status visibility, provenance indicators and direct links into

Intelligence/Product flows. Products show the naming product number + linked ClickUp id and the unit-economics fields (S18); Competitors show linked A13 entities; Research Intake shows promote/reject state; Rip Sources feed Expand.

Acceptance. `pnpm --filter web test -- workspace-batch3`; parity checklist signed by the list owners; one dual-run walkthrough attached.

Out of scope. Launches, Demand, Scripts, Creatives, Campaigns, Adsets, Email Flows, company-ops lists.

S17 · workos host move into the stack + credential discontinuation · S · L · S4–W6

Read first. S0a inventory; `service/.env.example` + `config.ts` (`EnvSchema`); Railway config; `scripts/register-clickup-webhook.mjs`; `rescale_service` role on ecomprofits.

Deliverables. `services/workos` deployed as service `workos` in the Skynet stack (Node 24, limits), env from `.env.production`; new ClickUp bot token, OpenRouter key, Discord token, AssemblyAI key, Google service account, ecomprofits `rescale_service` password — old values revoked (report); webhook re-registered to the new host; cron parity (reconcile, platform-sync, bounty, group-sync) plus the media proxy (`GET /media/:fileId`) verified against a 24-h Railway side-by-side, with an explicit parity check over every retained module (webhooks, reconcile, platform-sync, bounty, group-sync, media proxy, notification/rules); rollback = Railway kept warm for the rollback window then deleted; `docs/runbooks/S4-workos-host-cutover.md`.

Acceptance. Parity checklist green; revoked-credential report; Railway off after the window.

S19 · Final batch API + ClickUp read-only cutoff: funnelsLanders, ugcCreators, emailFlows, campaigns, adsets · S · L · S4–W7

Context. The final batch completes the replacement of the 13 creative-workflow lists and turns ClickUp into a read-only rollback surface for those lists only.

Read first. D42 (incl. the `ugcCreators` schema), S17 host-move runbook, the naming assemblers (`assemble.ts`, `funnelish-name.ts`, `klaviyo-name.ts`, `campaign-name.ts`), `platform-sync/`, *`modules/rules/`*.

Deliverables. Native stores + mirrors for the five lists (campaign/adset naming via the assembler; funnel availability + Funnelish/Klaviyo namers ported); then the 13 lists set read-only in ClickUp (statuses locked, a banner comment), the mirror kept as read-only rollback for the retention window; company-ops lists untouched; `M7` KPI + `M9` Rules parity and the Phase-8 workos features for the in-scope lists: launch outcomes propagation, fulfillment tracker, winner-iteration demand, URL rip engine, auto-CBO / increment fields, Discord/group-sync notification rules; the `ugcCreators` schema from D42 implemented here; `vitest services/workos/tests/board-batch4.spec.ts`.

Acceptance. Test green; read-only cutoff checklist; bounty still runs from the native stores.

Out of scope. Company-ops lists, payroll/bounty replacement, non-creative Workspace programs.

V26 · Final batch native UI + Workspace as the default path · V · L · S4–W7

Context. The last creative-workflow boards go native and Workspace becomes the default entry path; company-ops stays in ClickUp.

Read first. S19 API + mirror contracts; D42 board-authority doc (incl. the `ugcCreators` schema); the shell nav (V6).

Deliverables. Native boards for Funnels/Landers, UGC Creators, Email Flows, Campaigns and Adsets; nav default switched to native Workspace; company-ops deep links remain explicit ClickUp links; the read-only state of the retired ClickUp lists is visible in the UI.

Acceptance. `pnpm --filter web test -- workspace-batch4`; nav walkthrough attached showing the native default path and the ClickUp-only company-ops links.

Out of scope. Company-ops board replacement.

Bridge track B — standalone work for Mathew while S1 runs (temporary)

These cards live in the standalone repos (`meta-ads-scraper`, `RescaleOS`) and are retired by A19 / A20 / B0. Same standard as the slice cards. The PubMed R-cards are specified in `development-processes/pubmed-evidence-layer-plan.md` §9 (R1–R8) and are only summarised here.

R1 · PubMed doctrine file · Bian-side / Vince · S

Deliverables. `engine/skills/_doctrine/evidence-contract.md`: the evidence-class table from `pubmed-evidence-layer-plan.md` §2 (`strong / moderate / preclinical / case-report / narrative-review / traditional-use / unclassified`), the claims policy (evidence class ≠ permission; DSHEA gate; RAW → CA derivation; `form-mismatch` blocks claims), citation format, and transport order WebFetch → MCP → paste. **Acceptance.** File merged; referenced from `CLAUDE.md` routing.

R2 · RescaleOS `/evidence-layer` skill · Bian-side / Vince · M · depends R1

Deliverables. Skill in `engine/skills/evidence-layer/SKILL.md`: inputs mechanism, concern, ingredient(s), market-language seeds; six lenses (mechanism, root cause, symptoms, ingredients, failed solutions, desired outcomes) as PubMed E-utilities queries over WebFetch first; outputs M0–M9 (evidence table with class + citation, mechanism angles, fact-based hooks, authority statements, belief-shifting arguments, symptoms to call out, why other solutions fail, proof anchor, combine step with Reddit VOC); writes `evidence_report_{product}.md`; dry-run fixture. **Acceptance.** Fixture run classifies the sample studies correctly; the report is written in the workspace and consumed by the downstream skills.

R6 · Prove on one product · S · depends R2 — Acceptance. One living evidence report for the chosen product; retro edits applied to R1/R2.

R3 / R4 / R5 · Consumers · S each · depends R2, R6 — BPT Phase 2.5 reads the evidence output; belief rows cite evidence rows or are marked as non-evidence-derived; scriptwrite feeds from M2/M3 with RAW/CA handling. Acceptance. One BPT run, one offer brief and one script show cited evidence.

R8 · RescaleOS maps + memory flip · S · depends R2, R6 — routing, `supplements/CONTEXT.md`, `engine/memory/evidence-layer.md` say PubMed is live and is evidence, not VoC.

R7 · Console MCP wiring · Bian-side / Vince · S · depends R6 (optional accelerator)

Deliverables. `console/src/server/env.ts` (`getPubmedMcp()`), `console/src/server/agent/managedSession.ts` (`mcpServers.pubmed`, `mcp__pubmed__` pre-trust), `console/.env.docker` template, `DEPLOY.md`; wired to the standalone PubMed MCP when that

endpoint exists. **Acceptance.** A console session lists the four PubMed tools and `/evidence-layer` reports transport `mcp`.

R7a · PubMed scraper adapter + MCP · Agon · M · depends R6 (optional accelerator)

Deliverables. `lib/market-intel/pubmed-{client,normalize,classifier,worker,corpus}.cjs`, `lib/mcp/pubmed-mcp.cjs` at `/mcp/pubmed`, wired for RescaleOS use. **Acceptance.** The endpoint serves the four tools and produces the same report contract as WebFetch.

I1 · Scraper LFS bridge · Agon · L

Context. Mathew wants ranked long-copy native ads from an Ads Library link and continuous lander breakdowns before S2 Intelligence exists.

Read first. `scraper.js` (quick scrape, impression-rank order), `lib/deep-crawl/*`, `ads_browse` (`destination_url ILIKE` search), `lib/teardown/{worker,prompts}.cjs` (`teardown_model` setting), `lib/analysis/taxonomy.cjs` (format axis: no long-copy class), `lib/media-storage.cjs`, `tools/doc-render/` (self-contained HTML pattern).

Deliverables. (1) Long-copy detection as a bridge field `ads.is_long_copy_native` (heuristic: `format ∈ {Native Image, Static}` and `length(ad_copy) ≥ 900` chars, threshold in settings) — not a taxonomy change. (2) `POST /bridge/lfs/runs {ad_library_url, landers[], model}`: quick scrape + deep crawl of active ads → filter by landers on `destination_url` → rank (impression rank, then run span) → enqueue the new teardown tier `lfs_extract` (prompt: "extract and analyze the long-copy native ad form: structure, hooks, word choice, reading grade level, persuasion devices, rhythm, CTA pattern; produce a 'write one for me' packet"; `teardown_model` per run, default `x-ai/grok-4` via OpenRouter). (3) `GET /bridge/lfs/runs/:id` (status) and `GET /bridge/lfs/runs/:id/package`: a self-contained HTML report (ranked cards with image, copy, run span, destination, extraction) + per-ad JSON + media manifest. (4) Schedules: a run can be registered per brand + landers for daily re-execution.

Acceptance. One run from the Primal Queen Ads Library link with both landers returns the ranked candidates and an HTML report that opens standalone; Grok extraction present per selected ad and reproducible from stored inputs; `node --test test/` extended.

Out of scope. Native pages; taxonomy redesign; anything inside RescaleOS.

I2 · Primal Queen continuous job + export/drop · Agon · M · depends I1

Deliverables. Drop-target config (`BRIDGE_DROP_ROOT`: local path and/or Drive sync path); per run: `swipes/<brand>/<date>/` gets all candidates (HTML report + per-ad JSON + images),

`copy/<brand>/` gets drafted LFS packets; `winners/` and `losers/` only on an explicit human flag; stable names, re-runs idempotent per report scope; daily schedule registered for both landers.

Acceptance. Two consecutive daily runs produce the package in the target root with no duplicate ambiguity; Mathew opens the report from his folder.

Out of scope. Folders as canonical state; any auto winner/loser classification.

MC1 · RescaleOS model-configuration boundary · Bian-side / Vince · S

Deliverables. `docs` /settings state the true boundary: session model choice = Claude models via `setModel` (`sessions.model`, the usage-fallback chain); non-Claude models (Grok etc.) are reachable only through the scraper/OpenRouter tiers (`teardown_model`, `lfs_extract`) or a dedicated MCP; the `/lfs` skill exposes `--model` as the scraper tier's model, not the session's. **Acceptance.** No "any model" claim remains in RescaleOS docs; `/lfs --model x-ai/grok-4` sets the scraper run's model.

I3 · RescaleOS `/lfs <ad-library-url>` skill · Bian-side / Vince · M · depends I1, MC1

Read first. `engine/skills/npt/SKILL.md` (slash-command pattern), `engine/skills/_doctrine/`, I1 API, the accelerator folder convention.

Deliverables. `engine/skills/lfs/SKILL.md`: submit run (`--landers`, `--model`) → poll → fetch package → file into the workspace (`swipes/`, `copy/`) → print report path + top ranked ads → optional `--draft`: write one LFS using the stored extraction packet as the only source. No scraping, no Grok inside RescaleOS.

Acceptance. One end-to-end invocation from an Ads Library URL to filed artefacts; the draft cites the stored extraction.

I4 · Miguel's feedback → `rules.md` · Vince · S · depends Miguel's notes

Deliverables. `rules.md` in Mathew's structure: hard rejects · tendencies · before/after pairs · reaction-log format · promotion rule (one reaction → repeated pattern → reusable rule); sections for AI animated ads and cartoon ads; referenced by the relevant RescaleOS skills. **Acceptance.** File exists; at least one reaction → rule example.

B0 · Bridge sunset checklist · Agon · S · after A19 + A20 (Skynet card)

Deliverables. `docs/checklists/bridge-sunset.md` : every bridge path (I1 endpoints, I2 drops, `/lfs` , `/evidence-layer` transport notes, R7 MCP) either ported or removed, owner named, folder drops replaced by native entities, docs no longer describe the bridge as active. **Acceptance.** Checklist signed by Agon and Mathew.