Agentic Development Environment Technical Overview

A practical reference for reusable AI engineering workflows

Designed, integrated and operated by Zahidul Islam

Evidence checked Sep 7, 2026

Latest verified developments — 14 October 2026

This update comes first so the current capabilities can be read before the September baseline. It draws on the live project knowledge index, exact source sections, installed configuration, current local source and dated verification records. The original chapters, six tables, architecture image, links and incident accounts remain below. Their measurements retain their original dates; this review does not turn earlier tests into new live results.

Practical Claude Code, Codex and agentic engineering skills

The evidence now covers more than installing two coding assistants: provider-isolated CLI configuration; project-scoped recovery; custom MCP server implementation; typed tool contracts; bounded source retrieval; browser integration diagnosis; approval-bound automation; and release verification. These are concrete engineering capabilities that support describing advanced hands-on experience. No external certification, fixed experience duration or measured productivity multiplier is asserted.

Claude Code work includes the Windows executable launched from WSL, its separate configuration and authentication, reusable skills, a reviewer definition and an isolated gateway profile. Codex work includes the native Linux CLI, separate Windows configuration, MCP discovery, permission-aware checkpoints, provider compatibility fixes and recorded real-client workflow execution. Installing the same integration in one surface does not establish it in the others.

Current verification and corrected usage policy

On October 14, live knowledge synchronization and integrity audit completed with 22 active sources, 1,720,507 characters, 555 chunks and no errors, unsupported files or audit issues. These are pre-update counts. Eight knowledge-index tests passed. The 37 switcher tests also passed from a local source/test snapshot; this avoids fixture permission changes in the protected installation. The initial installed-suite attempt had two filesystem-permission errors, not established launcher defects.

Current policy treats usage percentages as informational. No percentage alone triggers stopping, committing, checkpointing or switching. A user request or an actual provider refusal determines the next step. The September chapter's 95% threshold and 80–90% suggestion are retained as historical instructions and are superseded here. Context, subscription allowance and OpenRouter spending remain separate; unavailable account meters must be reported as unavailable.

CLI operation, browser access and evidence discipline

Recovery and configuration as engineering boundaries

The launcher remains a deliberate Subscription/OpenRouter chooser. Checkpoints record the task, decisions, changed files, verification and remaining work inside the owning project's ignored memory folder. The successor must inspect current files and treat historical excerpts as context. Cross-provider native transcript resume remains unverified. September's real response and file-continuation tests remain dated evidence; no paid model inference was rerun for this update.

Provider endpoints, model choice, pricing and operational limits belong to central configuration; prompts and product copy belong to maintained data. Secrets stay in private stores and ignored recovery records. Configuration-owned model selection is distinct from authorizing paid use. Quality work requires explicit criteria, appropriate tests/types/lint/security/build checks, the real user flow and a recorded review verdict; written instructions alone do not prove those gates executed.

Browser integration: tested route and separate failure modes

September 21 records verify Microsoft's Playwright extension through the playwright-chrome MCP: 13 of 13 operations passed, including signed-in page reading, a separate test tab, three navigation/read cycles, screenshot inspection and cleanup preserving the original tab. This is dated functional evidence, not a long-run reliability or concurrent-browser benchmark. The October 14 check listed the current connection's Welcome tab; authenticated-page behavior was not rerun.

The official ChatGPT extension is a separate integration. Its runtime handshake and eventual tab discovery did not establish full content access: the native comparison hit a sandbox metadata validation error before JavaScript execution. The useful lesson is to separate transport, tool discovery, authentication, page read and permitted action rather than claiming that one successful initialization proves the complete browser workflow.

MCP authentication and schema diagnosis

October 13 records show an official MCP connection repaired through native OAuth, 30 tools discovered, and separately approved profile fields verified through independent readback while unrelated fields were preserved. The published connector still exposed stale action enums. Authentication health and tool-schema compatibility are therefore separate checks; a working native route does not prove every connector representation is current.

Project evidence is now located through a private verified folder directory, then checked against the owning project's instructions, dossier, relevant source and tests. Live knowledge sync/audit precede claim selection, and exact source sections establish the selected claims. Public documentation excludes private directory paths and recovery details. Existing Google Docs keep their IDs; exports are verified before canonical PDFs are replaced.

Applied MCP systems and bounded local agents

Fleetwright: durable browser workflows and local drafting

Fleetwright's custom stdio MCP exposes six bounded tools: workflow_contracts, submit_workflow, list_jobs, inspect_job, cancel_job and export_evidence. Current source and the live overview agree on tenant-bound grants, separate observation/execution scopes, current authorization checks and human-approved versioned workflow contracts. The model cannot grant approval. Lost acknowledgements retain UNKNOWN until matching destination receipts resolve the outcome; an absent receipt does not authorize another action.

Recorded actual Codex execution submitted, inspected and exported both owned workflows using the existing subscription login. The corresponding historical record explicitly defers actual Claude client verification. The latest October 13 release record reports 274 passed cases and six expected skips, with separate platform/installed-model evidence, quality/security gates, review and deployed artifact parity. These are that project's dated results, not tests rerun by this document update.

A separate standalone agent uses SQLite and an installed local Ollama model to draft typed text. Maintained templates own destinations and receipt contracts; the human reviews the exact draft and authorizes one hash-bound save. It requires no PostgreSQL, Redis, Docker or paid inference API at runtime. Dedicated browser profiles and loopback boundaries reduce exposure; the documentation explicitly distinguishes them from an operating-system sandbox.

Chronos and VitalProbe: different MCP boundaries

Chronos has an eight-tool local Python Meta integration for business context, reporting and durable before/after campaign proposals. A separate interactive owner CLI supplies approval. The 37 recorded local tests include actual stdio initialization and CLI behavior. Live Meta access and actual Claude attachment still require owner setup and verification. Writes are limited to creating paused awareness campaigns or pausing campaigns; budgets, activation and publishing remain outside scope.

VitalProbe uses MCP in two directions: an adapter tests an MCP tool as the target, and a licensed server exposes bounded suite execution, history, comparison and learning-draft operations. Its LangGraph example reuses the canonical test-patient result contract. This establishes reusable evaluation and tool-integration design without presenting test patients as production clinical validation.

Related orchestration work and source boundaries

Secure Hybrid Hub remains a separate experimental, open-source broker/CLI. Its latest overview cites September 24 evidence of 257 passing tests and one expected skip, while Phase 1 closure, real-client operation and future scale/availability objectives remain incomplete or unproven. The seven hybrid skills belong to that project; they are not the launcher backend. Third-party CLIs, models, plugins and scanners remain attributed to their upstream providers.

Sources: exact live Fleetwright, Chronos, VitalProbe and Secure Hybrid Hub sections; local MCP/agent READMEs and source; launcher verification; browser evidence; and official Codex MCP, skills and configuration documentation. Private evidence records retain exact source locations and checks.

Historical baseline — 7 September 2026

The original technical overview follows in its existing order. Its cover evidence date, tool inventory, test counts, provider configuration and usage guidance describe the September inspection. Use the October update above for current policy and newer capabilities; retain these chapters for architecture rationale, failure history and earlier verification.

This environment brings Claude Code, Codex, Google Workspace, project knowledge, reusable skills and security checks into a repeatable engineering workflow. Its central contribution is the integration work: choosing how agents start, which provider pays for a session, how unfinished work survives a switch, where project facts live, and what evidence supports a completion claim.

The working setup includes a subscription and OpenRouter chooser for both coding CLIs, project-local handoff checkpoints, a custom knowledge MCP backed by SQLite, service-account access for shared Google documents, common global instructions, and reusable review and security workflows. It combines custom tooling with third-party products; it does not claim authorship of Claude Code, Codex, their models, Google Workspace or the installed scanners.

This reference is intended for engineering reviewers, collaborators and clients who want to understand the system and the decisions behind it. It covers the configuration inspected for this edition and distinguishes current checks from earlier recorded tests. Credentials, account identifiers, private infrastructure addresses and personal filesystem paths are deliberately excluded.

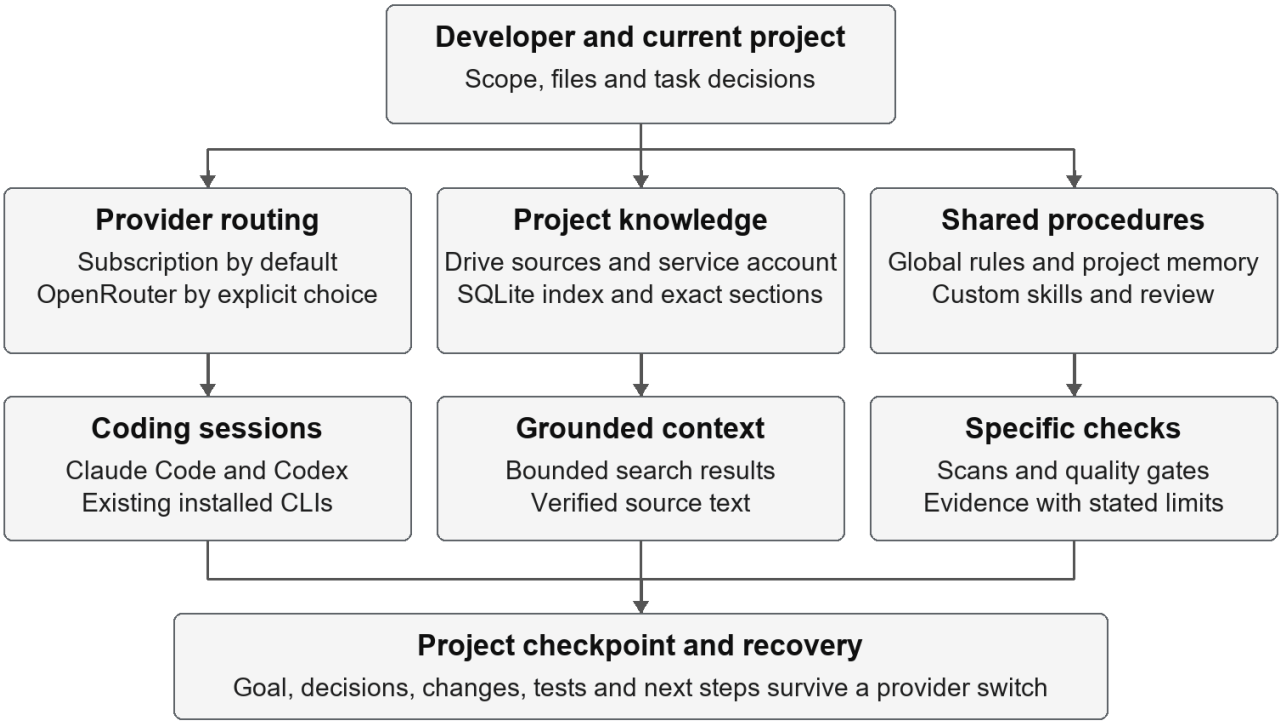
1 The engineering problem

An AI coding session depends on more than model quality. A capable model can still lose the thread when its subscription allowance runs out. A replacement session can repeat completed work, miss a previous decision, or act on the wrong project's notes. An expired Google credential can make a working knowledge workflow appear broken. A collection of installed tools does not, by itself, solve these problems.

The environment addresses those operational failures with explicit routing, recoverable task state, shared evidence and repeatable checks. The same user-facing CLI commands can start a subscription session or a deliberately selected OpenRouter session. Project checkpoints carry the goal, changes, test evidence and next steps to the successor. The knowledge service keeps complete source text available for exact retrieval while limiting how much enters each model response.

The value is visible in the workflows and their verification: continuing a small coding task across providers, refreshing and auditing the knowledge index, and testing the handling of missing keys, unsafe checkpoints and incompatible provider features. No measured productivity multiplier, universal reliability claim or comparison with other developers is asserted.

2 Architecture and responsibilities



The user's project remains the working context. The launch layer selects the access route; each CLI performs the coding work; skills describe repeatable procedures; local helpers validate and persist important state; and connected services supply approved external information. The developer retains decisions about paid access and the scope of the task.

Layer	Responsibility	Current implementation
Entry and routing	Select subscription or OpenRouter access	WSL Bash launchers and a Python chooser
Coding runtime	Run the interactive development session	Native Linux Codex and Windows Claude Code launched from WSL
Continuity	Save and recover the current project's task	Markdown checkpoints, request markers and a restart prompt
Knowledge	Refresh, search and verify project evidence	Custom career-knowledge MCP with SQLite FTS5
Google access	Read and edit shared documents; create user-owned outputs	Service-account Python helpers and an authenticated Drive connector

Layer	Responsibility	Current implementation
Working standards	Reuse instructions, skills and reviews	Global rules, custom skills and a reviewer definition
Security and quality	Detect defined issues and collect verification	Edit scanning, pre-commit checks, CI templates and test suites

These layers have different strengths. Instructions guide an agent's behavior. Python validators and scanners perform specific checks. A configured hook is not proof that every edit was scanned, and a successful test does not establish compatibility with every future model or plugin.

3 One entry workflow for two coding CLIs

From a project directory in WSL, the developer runs claude or codex. Both commands invoke the chooser. Pressing Enter selects the existing subscription access; selecting OpenRouter leads to a separate yes-or-no paid-session prompt. The launcher displays the configured model and dated token rates before starting that route.

The implementation preserves the installed coding binaries. It does not replace the user's subscription accounts with API authentication. Administrative commands such as help and version can bypass the interactive menu, while ordinary noninteractive launches are rejected when the launcher cannot establish a deliberate access choice. Raw CLI commands remain available for native automation and same-provider workflows.

An important platform detail is that a WSL terminal does not imply a Linux executable. Here, Codex runs as a native Linux program, while the Claude launcher invokes the existing Windows executable. That difference determines which configuration files, path forms and security bridges apply.

Surface or route	Instruction location by environment	Why it exists
Codex CLI in WSL	WSL Codex global instructions	Used by the native Linux CLI
Claude CLI started from WSL	Windows Claude global instructions	The launcher executes Windows Claude
Claude through OpenRouter	Isolated Claude configuration	Keeps gateway authentication separate from subscription credentials
Codex on the Windows desktop	Windows Codex global instructions	Supports Windows-hosted tasks
Native WSL Claude configuration	Additional identical copy	Not needed by the current Windows-backed Claude launcher

Five global instruction files were synchronized and hash-checked. They are separate copies, not an installed continuous synchronization service. Future rule changes must be propagated and verified again. Fresh interactive WSL terminals load the launchers automatically; manually sourcing the shell startup file is only needed when installing the functions into an already-open terminal.

4 Task continuity across a provider switch

The switch workflow transfers a task checkpoint, not a promise of byte-for-byte conversation compatibility. This distinction is essential: the successor receives a concise record of what matters and can inspect the working tree, without depending on one provider understanding another provider's entire session format.

1. The developer asks to switch while the current agent can still respond.
2. The switch-provider skill records the goal, constraints, decisions, changed files, verification evidence and remaining work.
3. A helper verifies that the checkpoint belongs to the current project, is ignored by Git, and passes the checkpoint secret scan.
4. The helper queues a project-local request marker. The developer exits the CLI.
5. The launcher shows the Subscription/OpenRouter menu again.
6. The selected successor session receives an automatic prompt to read the saved checkpoint before continuing.

The developer does not need to paste that recovery prompt. Its automatic appearance is intentional. The prompt tells the successor to treat historical excerpts as context, inspect current files, preserve unrelated work and ask when a necessary next step is unknown.

If the subscription is already exhausted, ai-switch claude or ai-switch codex can use the most recent checkpoint or accept a short user-written recovery note. The local helper can record Git state without making a model request. It cannot reconstruct decisions and test results that were never saved, so missing information is explicitly marked as unknown.

Checkpoints remain inside the current project's ignored memory/ directory. Request markers were also moved there so they are writable from the Codex workspace sandbox. Ordinary launcher startup works outside Git, but private switch checkpoints currently require a Git project so ignore status can be verified. Native cross-provider resume and fork options are deliberately not used by this workflow.

5 Billing and usage signals

Three measurements must remain separate: conversation context, subscription allowance, and OpenRouter spending. A context percentage estimates how much conversation fits in the model window. A subscription meter reports a provider's account allowance. OpenRouter usage measures charges against the configured key and account credits.

The ai-status helper reports verified OpenRouter key usage, the key cap, remaining cap and reset period. Remaining key cap is not necessarily the account's spendable balance. The same account credits support both coding clients; creating a second paid account is unnecessary for these routes.

The inspected setup used a total key cap of $20. A total cap is cumulative: adding account credits does not, by itself, increase that cap. Any future top-up or cap change is a separate user decision. The launcher does not purchase credits, turn on automatic top-up or alter the provider's extra-usage settings.

Claude's subscription workflow has a configured 95% checkpoint threshold based on its five-hour usage signal. That signal must not be reused for Codex or OpenRouter. The launcher itself does not continuously monitor subscription percentages or instantly display a menu at a threshold. An early user-requested switch at 80–90% can use the same checkpoint workflow, provided the percentage comes from a verified provider meter.

The gateway model is configuration-owned. The recorded live test used meta/muse-spark-1.3; this is a compatibility result for that configuration, not a recommendation that it is the cheapest or best model. Provider pricing, model availability and feature support must be checked when the configuration changes.

6 The custom knowledge MCP

The career-knowledge service exposes project evidence through seven bounded operations. Its sources are a recursively watched Google Drive knowledge folder and a local portfolio configuration. The source document text is preserved in a private SQLite database, with chunk boundaries and exact character ranges that support later inspection.

Incremental synchronization exports new or changed documents and leaves unchanged ones alone. Removed documents are marked inactive while retained privately, so historical evidence is not silently destroyed. SQLite FTS5 and BM25 provide lexical retrieval. This particular index does not require paid model inference to synchronize or search, and should not be described as an embedding-based semantic index.

MCP operation	Purpose
career_sync	Discover source changes and refresh the index
career_status	Report index and synchronization status
career_inventory	List represented source documents
career_audit	Check indexing integrity and coverage
career_search	Return bounded candidate excerpts
career_match_requirements	Find candidates for a set of requirements
career_get_document_section	Retrieve the exact source section for verification

The workflow separates finding a plausible excerpt from proving a claim. Search results are candidates. Before using a metric, capability or project outcome in a proposal or portfolio statement, the agent must inspect the exact source section and preserve its qualifications. A keyword match does not prove production status, experience duration or a completed deployment.

At the start of this documentation pass, synchronization and audit reported 20 active sources, 1,190,969 characters and 382 chunks, with zero errors, zero unsupported files and zero audit issues. These are a dated snapshot before adding this overview, not permanent totals. Eight knowledge-index unit tests were rerun successfully.

The index's text boundary matters. Text existing only inside images, drawings, comments or unresolved suggestions is not established by ordinary document text export. Complete index coverage means complete coverage of its supported, configured text sources; it does not mean every Google Workspace object or every document on the computer was inspected.

7 Google Workspace access and service accounts

The machine has GWS CLI installed, with the inspected executable reporting version 0.22.5. Its own version output states that it is not an officially supported Google product. The broader integration also uses Python Google API clients and a connected Google Drive/Docs tool surface. These are distinct access paths rather than one interchangeable authentication store.

The Python knowledge provider and document-update helper prefer a configured service-account credential file when present. They retain OAuth as a fallback when it is absent. Service-account access is limited by the permissions on the documents and folders shared with that identity; possessing a key does not grant access to every file in the user's Drive.

The migration notes record repeated invalid_grant failures in the previous OAuth arrangement. Refreshing one token store did not necessarily repair another. Google documents a seven-day refresh-token lifetime for external apps in Testing status when their scopes extend beyond basic identity. Moving the shared-document workload to a service account removed that specific interactive reauthorization dependency. Google OAuth documentation

That is not an assertion of permanent authentication. Access tokens still have lifetimes, permissions can change, and service-account keys can be revoked or rotated. Source comments that describe a key as never expiring should not be interpreted as a guarantee of indefinite access.

File creation has a separate ownership boundary. Service accounts have no personal Drive storage quota and cannot own files. Existing shared documents can be read or edited when authorized; new user-owned documents should be created through an authenticated user route. Organizational shared drives are a different ownership model. Google Drive ownership documentation

The document lifecycle therefore keeps a stable Google Doc for continuing updates and exports a PDF after content and layout verification. A simple whole-body replacement helper exists, but it should not be treated as a formatting-preserving editor for every structured document. Native edits or a verified import are appropriate when headings, tables, images or other document structure matter.

8 Shared rules and project memory

Global instructions cover research, verified claims, paid actions, configuration hygiene, local-first deployment, testing, security and recovery. The merged usage section explicitly distinguishes Claude subscription behavior, Codex's verified usage signals and OpenRouter spending. Common text reduces avoidable differences when the same project is opened in another agent.

Project-local memory serves a different role from global rules. The memory/ folder holds working notes and handoff checkpoints; MEMORY.md indexes those notes; and PROJECT-DOSSIER.md records durable architecture, evidence and incident history. Public overviews are scrubbed outputs derived from that private record. The intent is to keep project knowledge with the project when folders move or sessions change.

Where an agent expects a hardcoded memory location, a configured path or filesystem link can preserve compatibility while keeping the actual notes in the project. Such links must be checked after moving a folder. A copied index pointing at an old location is not a successful migration.

The environment also maintains private credential recovery records. Gitignore prevents accidental staging in normal workflows, but it is not encryption or access control. Those recovery records must remain outside public documentation and be handled as sensitive local material. This overview is not an audit of every credential store or backup on the machine.

9 Custom skills and review procedures

Ten distinct custom skill definitions were located across the global environment and inspected project directories. Mirrored Claude and Codex versions are counted once. This inventory establishes that the definitions exist and describes their purpose; it does not claim every procedure was executed during this documentation pass.

Skill or group	Purpose and scope
production-ready-code	Global acceptance criteria, smallest-scope implementation, quality gates, real-flow verification and review
switch-provider	Global provider handoff with a private project checkpoint
hybrid-init	Register and discover an approved project in the separate Hybrid Hub
hybrid-run	Submit and follow broker-controlled implementation work
hybrid-status	Read task state and policy-safe evidence
hybrid-resume	Resume after the missing input or approval is supplied
hybrid-cancel	Cancel while preserving recoverable work and evidence

Skill or group	Purpose and scope
hybrid-dossier	Query approved project facts or propose a typed change
hybrid-promote	Request controlled promotion through the broker
compliance-gate	VitalProbe-specific scope checks and verification workflow

A prod-code-reviewer agent definition complements the production workflow. Its purpose is to review against criteria in a context that did not write the implementation. A review file being installed is not proof that review happened; the actual verdict belongs with the task's evidence.

The seven Hybrid Hub skills belong to a separate project. Its live overview records an incomplete Phase 1 and test-data verification at its earlier reconciliation point. Those definitions are relevant examples of custom orchestration work, but they are not evidence that the current provider switcher runs through that broker or that the broker has completed production validation.

10 Third party tools and MCP integration

Custom integration and third-party tooling are presented separately so the engineering contribution is clear. The installed frontend-design plugin, official documentation MCP, LinkedIn integration, desktop connectors and built-in artifact skills extend the environment, while their upstream implementations remain third-party work.

Observed integration	What the inspection establishes
Project career-knowledge MCP	Custom server configuration and seven-operation source implementation
OpenAI documentation MCP in WSL Codex	Configured documentation integration
LinkedIn MCP in Windows configurations	Configured third-party integration; live account behavior not tested here
Node REPL in Windows Codex	Configured execution integration
Frontend-design plugin in Claude	Installed and enabled third-party plugin
Google Drive and Docs connector in this desktop session	Live search and document-read operations succeeded

Capabilities vary by surface. Installing an integration in Windows does not automatically install it in WSL. A plugin available in the desktop app is not automatically available to every CLI or provider route. In the

tested Codex OpenRouter profile, ChatGPT app connectors are disabled because the selected provider rejected an advertised tool name. The subscription route retains its own app configuration.

11 Security and quality controls

The setup combines guidance, hooks, repository checks and explicit task verification. The global edit scanner understands supported Claude edit events and Codex patch payloads, extracts file paths, and runs Gitleaks and Bandit. A WSL bridge supports the native Linux Codex environment. The scanner redacts secret findings rather than printing credential values.

Repository templates add pre-commit checks and a CI workflow. The available scanners include Gitleaks, Bandit and Semgrep. A paid Claude CI review job is optional and remains commented out in the template; the presence of the template is not proof that every project has installed hooks or enabled CI.

The general edit hook has important limits. Its source and README show fail-open behavior on internal errors or unavailable tools, deliberate skips for private credential files and selected directories, a large-file cutoff, and an existing Bandit B601 exclusion. A quiet hook therefore does not establish that every file was scanned or that the application is secure.

The provider handoff has a stricter boundary: checkpoint scanning failures block the handoff. Regression tests cover a detected local secret and scanner failure. That check helps prevent unsafe checkpoint transfer, but should not be expanded into a claim that all possible confidential information can be detected automatically.

The production workflow asks for acceptance criteria before implementation, appropriate tests and static checks, real-flow verification and a fresh-context review. These are repeatable procedures. Their value depends on executing and reporting the gates for the actual change rather than assuming the instructions enforce themselves.

12 Verification and observed results

The documentation pass reran local tests and live knowledge reads. It did not rerun paid model inference, purchase credits, enable integrations, deploy infrastructure or test every installed plugin. Earlier provider tests are retained as recorded evidence with their original scope.

Check	Evidence and result
Switcher regression suite	WSL unittest discovery completed successfully; the suite's verification record lists 37 tests
Knowledge unit suite	8 tests rerun and passed
Knowledge sync and audit	Both complete; no reported errors, unsupported files or audit issues
Global instruction consistency	Five copies previously hash-matched; Windows copies rechecked during this pass

Check	Evidence and result
Claude subscription to OpenRouter	Earlier same-day checkpoint recovery, response, file edit and independent readback recorded as passing
Codex subscription to OpenRouter	Earlier same-day checkpoint recovery, response, file edit and independent readback recorded as passing
OpenRouter to subscription	Same-day user-run two-step continuity tests recorded in both CLI flows
Fresh WSL startup	Existing verification records login and non-login interactive shell checks
Native edit-hook invocation	Not independently established for every edit or surface by this document

The recorded model tests used the exact prompt hi openrouter and a two-step file task. One stage created the first line; after the switch, the successor recovered the checkpoint, appended the requested second line and read the file back. This is stronger evidence of useful continuity than receiving a greeting alone.

The earlier switcher verification also records passing type checks, lint, security gates and compilation for its reviewed source. Those are dated component results, not a claim that this documentation pass repeated every gate across all projects.

13 Failures and the engineering lessons

Wrong login flow after an OpenRouter refusal

Claude displayed a generic login suggestion after OpenRouter returned a model-access prerequisite. Logging into the client subscription did not address the service returning the error. The user completed the prerequisite in OpenRouter, and subsequent real requests succeeded. The lesson is to identify the failing provider and response before changing authentication.

Handoff state outside the writable sandbox

A checkpoint request marker in a home configuration directory was not writable from the Codex workspace sandbox. Moving the marker beside the ignored project checkpoint aligned the storage location with the worker's permissions. The lesson is to design recovery state around the actual execution boundary, including Windows-to-WSL path translation.

A compatible endpoint with an incompatible feature

The chosen provider rejected an application tool name advertised by Codex. The fix was scoped to the gateway profile rather than removing subscription integrations globally. The lesson is to test the full client request shape, including tool metadata, rather than treating endpoint compatibility as complete feature compatibility.

OAuth repaired in one place but broken in another

Separate Google authentication stores and a short Testing-mode refresh-token lifetime caused recurring failures. Shared-document access moved to a service account, while creation retained a user-owned route. The lesson is to separate credential lifetime, permission scope and file ownership when diagnosing an integration.

Global instructions that stopped matching

Independent Codex, Claude and gateway configuration locations accumulated different guidance. A merged provider-aware section and hash-verified copies restored consistency. The lesson is that synchronization needs a repeatable operation; identical files today do not ensure identical files after the next edit.

14 Boundaries and maintenance

The current workflow supports deliberate switching and task recovery. It does not promise an automatic switch before every quota failure, full cross-provider transcript compatibility, universal plugin support, a permanent Google credential, or comprehensive security enforcement. Those limits are part of the design record rather than hidden qualifications.

Maintenance should follow the ownership of each component. Model and provider values belong in the switcher's configuration. Installation paths belong in deployment configuration. Global rule changes require synchronization and hash checks. New knowledge sources require discovery and audit. A new custom skill requires a verified procedure, and a newly configured MCP needs a real connectivity and behavior check before being described as working.

The public documentation follows the existing project-view convention: a maintained technical overview, a stable Google Doc identifier, an architecture image and a PDF exported from the finished Google Doc. Future updates should revise that same document, check its native content and exported layout, and replace the local PDF with the new verified export.

15 Source basis

This edition draws on the live Career Knowledge Base, the separate Hybrid Hub overview, the switcher source and verification record, the knowledge MCP implementation and tests, the Google authentication loaders, custom skill definitions, global instructions, scanner source and installed configuration metadata. Exact local source paths and recovery details remain in private project records.

The strongest claims are supported by executable behavior or direct source inspection: deliberate provider selection, project-local checkpoints, successful knowledge synchronization, integrity auditing and passing local tests. The supporting integrations and procedural controls are identified at their actual verified level. Together they prove practical AI development environment engineering without treating installed tools or written policies as proof of outcomes they have not yet established.