LLM ENGINEERING

LLM Adaptation and Runtime

Data, Post-Training, Inference, and Model Change

WORKFLOW	EVIDENCE	PRODUCTION	OWNERSHIP

Komal Nakrani

First edition - Version 1.0.0

Copyright and professional boundary

Professional boundary and disclaimer

This book is technical and professional education. It is not legal, privacy, security, safety, accessibility, medical, financial, regulatory, audit, licensing, or compliance advice. A model card is not approval. A checksum is not supply-chain assurance. A simulated training curve is not a trained checkpoint. A passing local test is not production evidence. Readers must involve qualified specialists and explicitly designated organizational authorities when a decision requires their expertise or authority.

The LLM Engineer owns measured language-model-system behavior within delegated scope: preserve a behavior contract, diagnose a change-sensitive gap, construct traceable evidence, run bounded adaptation and runtime investigations, and recommend a disposition. The role does not acquire product, domain, data-rights, privacy, security, safety, legal, infrastructure, procurement, risk, or release authority by preparing the dossier.

The learning packs shipped with the volume are marked `NOT FOR LIVE CERTIFICATION BANK`. They support practice and review. Buying, reading, or completing this book is not required for any independent Abhyaas certification, and the book does not claim to confer certification.

Preface

Changing weights is easy to describe and difficult to justify. A recurring failure can originate in the task contract, data population, prompt, template, tokenizer, retrieval path, output validator, judge, runtime, or product workflow. Training before locating that layer can produce a costly artifact that changes nothing important or damages behavior the team forgot to protect.

LLM adaptation and runtime engineering treats weights, data, optimization, packaging, inference, and operations as one evidence-controlled system. The system begins from a frozen behavior baseline and asks a narrow question: what is the smallest intervention that could plausibly change the diagnosed residual without breaking the contract? Every later claim must retain the exact model, tokenizer, template, data, objective, evaluator, runtime, resource, and authority identity that made it interpretable.

This volume follows one recursive evidence loop:

1. reconstruct the Volume 1 behavior contract, baseline, evaluation, budgets, and unresolved gap;
2. inspect model, tokenizer, template, precision, checkpoint, and runtime compatibility before choosing a method;
3. compare no-change, system repair, supervised adaptation, PEFT, preference optimization, advanced methods, and replacement on mechanism and evidence;
4. specify rights, purpose, population, provenance, quality, exclusion, retention, and deletion behavior before constructing data;

5. separate train, development, evaluation, retention, and control evidence while making overlap and contamination limits visible;
6. freeze the objective, seeds, optimizer state, precision, effective batch, checkpoints, resource envelope, behavioral hooks, and recovery plan;
7. distinguish optimization loss and proxy preference from held-out target, retention, language, control, and resource behavior;
8. package weights, tokenizer, template, adapter, schema, runtime, configuration, lineage, licenses, checksums, and rollback as one compatibility-bound identity;
9. select inference and serving configurations on a measured behavior-resource frontier rather than throughput alone;
10. release only through named authority, bounded exposure, privacy-minimized signals, diagnosis, rollback, and complete change replay.

The loop permits rejection. A data recipe may reveal insufficient rights or coverage. A compatibility inspection may stop before a batch. A no-tune baseline may outperform a complex candidate on protected behavior. A smaller runtime intervention may dominate a weight change. A complete evidence process can end in hold without manufacturing a checkpoint or release.

The running system

The reader continues Mosaic Desk, a fictional appliance-support workflow assistant. Volume 1 established a typed, citation-bearing proposal path with authorized retrieval, explicit evidence state, uncertainty, abstention, escalation, deterministic validation, and a human decision gate before any external effect.

Volume 2 investigates a bounded synthetic residual: stable appliance shorthand and two lower-resource-language formatting segments remain difficult after credible prompt, context, schema, retrieval, and evaluator repairs. The proposed open-weight path is initially blocked by template and padding compatibility gaps. The chapters preserve that block. They specify data and experiments, compare deterministic simulated candidates, reject mismatched advanced methods, bind an incomplete package, model workload-specific capacity, and conduct a tabletop release/change incident. No real training, model execution, benchmark, artifact creation, deployment, or production outcome is claimed.

Humans retain warranty authorization and safety-critical repair decisions. Product and domain owners decide value and acceptable behavior. Data owners and privacy/legal authorities decide whether examples may be used and retained. Security owns threat acceptance. Platform and SRE own infrastructure, capacity, reliability, and recovery. Release authority decides exposure. The LLM Engineer makes the behavior and change evidence legible without absorbing those decisions.

Who this volume is for

The primary reader is a software, ML, data, evaluation, platform, performance, research, or product engineer who already understands the Volume 1 behavior-system discipline or can reconstruct the supplied frozen baseline dossier. You should be comfortable with Python or comparable programming,

schemas, version control, tests, structured logs, ordinary supervised-learning vocabulary, train/development/test separation, introductory probability, linear algebra, optimization, and ML evaluation.

This volume does not replace a deep-learning course, distributed-training specialization, accelerator-performance curriculum, privacy or security review, licensing analysis, or production SRE practice. Appendix A refreshes only the mathematics and measurement needed to interpret the decisions. Appendix E refreshes runtime concepts at task-decision depth. Escalation to specialists is an intended capability, not a failure.

How the volume is organized

The five parts follow the adaptation and runtime evidence loop.

- Part I earns the right to consider a weight change through a frozen baseline, compatibility inspection, and smallest-intervention decision.
- Part II engineers authorized learning evidence through a reproducible recipe, separation, instruction data, preference data, and protected retention/control cases.
- Part III runs bounded post-training investigations across experiment machinery, supervised adaptation, PEFT, preference optimization, and advanced-method rejection.
- Part IV binds the model-system package and qualifies its workload-specific inference and serving envelope.
- Part V operates the adapted-model dossier through release review, diagnosis, rollback, requalification, and change.

Each chapter has four layers. The main argument explains a decision and its tradeoffs. Mosaic Desk advances one durable dossier. Figures make boundaries, paths, and states visible without replacing the surrounding prose. The deterministic companion turns selected contracts into local fixtures and tests while retaining `trainingExecuted`, `modelExecuted`, `benchmarkExecuted`, and artifact-creation limits where applicable.

The seven appendices serve different jobs:

- Appendix A refreshes optimization, adaptation, uncertainty, and behavioral measurement concepts.
- Appendix B defines model, tokenizer, template, data, checkpoint, adapter, runtime, and package identity.
- Appendix C supplies copyable Mosaic adaptation/runtime artifacts and review gates.
- Appendix D explains the provider-neutral companion and safe extension rules.
- Appendix E refreshes inference, capacity, caching, queueing, and distributed-systems boundaries.
- Appendix F defines canonical terminology and adjacent-role boundaries.
- Appendix G explains source, claim, figure, edition, errata, and proof records.

Reading paths

For a complete first read, follow the chapter order. Later experiments are interpretable only because earlier identity, data, evaluation, and authority records remain frozen.

For adaptation triage, read Chapters 1-3 together. Do not choose a method until the residual, artifact tuple, smaller interventions, access limits, stopping rules, and disconfirmation evidence are explicit.

For learning data, read Chapters 4-8 in order. A large dataset is not defensible when rights, purpose, population, duplicate families, split isolation, template rendering, rater bias, retention, and control cases remain implicit.

For post-training, read Chapters 9-13 with Appendix A. Treat the included SFT, PEFT, preference, distillation, and continued-pretraining values as synthetic mechanics fixtures. The chapters teach comparison and rejection, not a recipe that predicts a real candidate.

For packaging and serving, read Chapters 14-16 with Appendices B and E. Integrity and compatibility are different gates. A capacity model is workload-specific. A fast variant must replay protected behavior before it enters a release packet.

For lifecycle and leadership, read Chapter 17 with the complete dossier. The chapter ends in hold because the package is incomplete. Its incident and migration paths are tabletop evidence, not a disguised deployment claim.

Working with evidence states

Use bounded evidence language in notes and reviews:

- `observed`: recorded by the named method for a named artifact, population, segment, runtime, and time;
- `inferred`: a reasoned interpretation that remains separable from the observation;
- `synthetic`: constructed teaching or test evidence that cannot establish a real-world outcome;
- `simulated`: a deterministic comparison that does not imply model execution or artifact creation;
- `supported`: current evidence is adequate for the stated narrow claim;
- `unsupported`: the claim exceeds current evidence or population coverage;
- `unknown`: necessary evidence is absent or cannot yet be reconciled;
- `untestable`: the current method cannot credibly evaluate the claim;
- `disconfirmed`: evidence contradicts the working hypothesis;
- `blocked`: a named prerequisite prevents the next transition;
- `retain`, `revise`, `reject`, `scope`, `hold`, and `release review`: explicit dispositions rather than confidence adjectives.

A useful adaptation claim names the behavior contract, residual, artifact tuple, data recipe, population, segment, method, version, evaluator, runtime, resource measure, limitation, owner, authority, and next trigger. A lower loss does not establish better behavior. A checksum does not establish compatibility. Passing aggregate behavior does not erase a protected-segment failure.

Running the companion

From the repository root, use:

```
node --test content/publications/llm-adaptation-and-runtime/companion/tests/*.test.mjs
node content/publications/llm-adaptation-and-runtime/companion/run.mjs
```

The companion uses Node.js built-ins, local files, deterministic fixtures, and no provider secret or accelerator. Do not paste customer data, secrets, raw production traces, proprietary training examples, or restricted model artifacts into it. Do not add a real training or provider call merely to make the example feel realistic. Any extension must preserve the same behavior contract, artifact identities, protected evaluation, authority gates, and failure oracle.

Figures and accessible use

The edition's figure sequence is V2-F01.1 through V2-F17.2: 34 original synthetic ImageGen PNG teaching illustrations. Layout, lanes, gates, locks, shapes, arrows, textures, and short labels supplement color. Read each image with its caption, nearby argument, and long description.

The figures explain relationships and decision states. They do not report measured training, model, product, user, capacity, cost, or production outcomes. The canonical sources are PNG. Delivery tooling may create derivatives, but derivatives are not the provenance source. Appendix G explains figure identity, dimensions, hash records, mirrors, corrections, and review status.

Sources, claims, and corrections

Material chapter claims use stable claim IDs that resolve through `claims.json` to `sources.json`. Sources include primary papers, standards, official documentation, model and dataset disclosures, and bounded first-party technical evidence. A source may support one narrow statement without proving Mosaic's population, rights, infrastructure, hyperparameters, capacity, or release fitness. Vendor evidence remains vendor evidence. A public recipe is not a transferable authorization or result.

This first edition was published on 2026-08-17 after complete web and PDF review. The canonical PDF is enabled for version 1.0.0. Publication records the accepted educational edition; it does not convert its synthetic examples into production evidence or transfer any product, domain, data, privacy, security, safety, platform, risk, or release authority.

Corrections must enter `errata.json` or a reviewed edition change. A correction identifies edition, location, problem, disposition, resolution, and replacement text when applicable. No web page or PDF should silently change while retaining the same released edition identity.

A note on judgment

Adaptation systems invite shortcuts: an architecture feature becomes a quality prediction, a token loss becomes a behavior claim, parameter efficiency becomes product fitness, a data license becomes sufficient purpose authority, a hash becomes compatibility, throughput becomes capacity, and a completed package becomes release approval. This volume refuses those substitutions.

The discipline is not opposition to adaptation. It is the ability to change weights and runtimes when justified while keeping the contract, evidence, regressions, resources, ownership, recovery, and formal authority inspectable to the people responsible for the consequences.

Contents

Part I - Earn the Right to Change Weights

The first adaptation failure often happens before training. A team sees a recurring output error, chooses a familiar fine-tuning method, and starts collecting examples without proving that the residual survives prompt, context, retrieval, schema, evaluator, or runtime repair. The resulting experiment may optimize an accidental interface or produce evidence that cannot be compared with the original system.

Part I establishes the adaptation gate. Chapter 1 reconstructs the frozen Volume 1 behavior baseline, separates inherited holds from a new synthetic shorthand residual, and writes falsifiable target, retention, budget, and rejection criteria. Chapter 2 inspects the exact weights, tokenizer, template, precision, checkpoint, and runtime tuple and stops on compatibility gaps rather than predicting behavior from architecture. Chapter 3 maps each residual to the smallest plausible intervention and compares no-change, system repair, SFT, PEFT, preference optimization, advanced methods, and replacement under evidence, access, resources, reversibility, and authority.

Mosaic Desk advances into `MD-09`: a frozen adaptation baseline, model/tokenizer inspection, and adaptation ladder. The open-weight candidate remains blocked by a training/serving template mismatch and unresolved padding assumptions. Data planning may proceed conditionally; training may not.

The handoff into Part II is a narrow adaptation hypothesis with explicit stop rules, not a method authorization. The next part must prove that examples are lawful for the purpose, representative of the target, reproducibly transformed, separated from evaluation, and protected against silent retention or control regressions.

1. Start From a Frozen Behavior Baseline

Turn a negative adaptation referral into a falsifiable investigation anchored to an exact, replayable behavior baseline.

Adaptation begins with a refusal to guess

Volume 1 ended with a useful negative result. Mosaic Desk's current release unit is held for named-authority review. A candidate provider migration is held while the qualified fallback remains available. A revoked-source defect belongs to retrieval authorization, not to model adaptation. Most important, `MD-08` labels the adaptation referral `not-justified`: the observed Gujarati citation regression is neither stable nor shown to be valuable, data-supported, or sensitive to a weight change.

That handoff is not permission to tune. It is the first input to an audit.

The fictional team now receives a second signal: several authorized internal cases contain appliance-domain shorthand mixed with structured Gujarati and English fields. Some outputs repeatedly normalize the shorthand incorrectly. Before anyone proposes training, the team must reconstruct the exact baseline, separate malformed inputs from genuine residual errors, and write a hypothesis that could be disproved.

The companion remains deterministic teaching material. It sends no provider request, trains no model, processes no customer record, and reports no product outcome. Its synthetic errors demonstrate how to preserve evidence boundaries.

Reconstruct the inherited state exactly

Start by verifying the artifact that crossed the volume boundary. The `MD-09` audit records the path and SHA-256 digest of `mosaic-migration-handoff.json`, plus the accepted Volume 1 verification digest. It copies the four dispositions without improving their language:

- release: `hold-for-authority-review`;
- migration: `hold-and-keep-current-fallback`;
- retrieval authorization: separate containment and repair;
- adaptation: `not-justified`;
- Volume 2 entry: `audit-required-no-adaptation-approved`.

If the handoff digest does not match, stop. If a referenced baseline dependency is missing, stop. If the case set or evaluator changed without a new identity, stop. An adaptation claim is interpretable only against a frozen behavior/configuration/evaluation baseline. [CLM-001]

The baseline is larger than a checkpoint. Record the task and authority contract, provider-neutral request/response contract, managed model identifier or open-weight checkpoint digest, tokenizer and template identity, decoding controls, retrieval corpus and policy, assembler, schema, case-set version, judge and rubric, runtime, environment, and resource budgets. Preserve raw case transitions and protected-segment outcomes rather than only an aggregate.

Managed and open-weight routes expose different evidence. A managed route may disclose a dated model identifier and API options while hiding tokenizer, weights, kernels, and serving topology. Record that opacity; do not invent hashes. An open-weight route must bind weights, revision, tokenizer files, template, quantization, runtime, dependencies, hardware class, and serving configuration. Platform/SRE owns availability and capacity operation. LLM engineering owns behavior-facing identity and replay evidence. Neither route changes product, domain, privacy, security, or release authority.

V2-F01.1 - Freeze the comparison unit before adaptation. Essential labels: frozen, candidate, contract, config, eval, budgets, controls. Shape and lock symbols supplement color. The figure supports `CLM-001`; it does not claim the baseline is production-approved.

Text alternative: A frozen baseline vault separates an identified baseline from a candidate and locks contract, configuration, evaluation, budgets, and controls before comparison.

Define a residual, not a mood

"The model is weak at domain language" is not a hypothesis. It names no population, mechanism, evidence, or rejection condition. Build a case-level residual ledger instead.

For each failure, record input provenance, authorized language and appliance segment, expected terminal state, actual terminal state, criterion, consequence, reproducibility, layer trace, and candidate explanation. Then test upstream surfaces in order: input validation, language-code mapping, task routing, source eligibility, retrieval, context assembly, message template, schema, evaluator, and runtime. A residual enters adaptation consideration only after identified simpler repairs fail to explain it.

The failure injection is deliberately adversarial to tuning enthusiasm. Half of the apparent multilingual model failures carry malformed upstream language codes. The input says Gujarati-English mixed, but a stale mapper routes it as English-only. Correcting the mapper makes those cases replay correctly without changing model weights. Calling that improvement a fine-tuning opportunity would corrupt the causal record.

The remaining constructed cases share a narrower pattern: authorized appliance shorthand must be expanded into a bounded typed proposal, and the same incorrect mapping recurs under the frozen template, retrieval evidence, decoding, and evaluator. That recurrence is still not evidence that tuning will work. It merely supports writing a testable adaptation hypothesis.

Use this form:

> For the frozen shorthand-and-structured-multilingual segment, the identified baseline repeatedly maps a bounded set of authorized abbreviations to the wrong typed field after input routing, retrieval, template, schema, and evaluator controls pass. A supervised intervention trained on authorized paired examples may reduce that error while retaining citation, abstention, conflict-escalation, schema, and resource gates. Reject the hypothesis if the residual disappears under a simpler repair, the data do not represent the target distinction, the candidate fails the target criterion, or any protected gate regresses.

Every noun points to an artifact. "Repeatedly" points to case/run evidence. "May" refuses to predict a result. The rejection clauses prevent training effort from becoming its own justification.

Pre-register what success cannot erase

Target gains, retained capabilities, controls, resource budgets, and rejection criteria should precede training. [CLM-002] This ordering matters because post hoc criteria can turn any noisy change into a win.

For Mosaic Desk, the target is a reduction in the specified shorthand-mapping error on a frozen, family-separated target slice. Retention includes English and Gujarati citation behavior, correct abstention when evidence is absent, conflict escalation, typed schema validity, source authorization, and the no-effect boundary. Controls include the unchanged no-adaptation baseline, a corrected-language-code baseline, a prompt/context repair candidate, and an evaluation replay with frozen judges.

Budgets cover authorized data volume, engineering time, compute envelope, artifact storage, evaluation runs, runtime memory, latency tails, and rollback complexity. The numbers in the companion are declared teaching budgets, not capacity estimates. Product and domain owners decide whether the target is valuable. Privacy and legal authorities decide whether data may be used and retained. Security owns threat acceptance. Platform/SRE owns infrastructure limits. Release authority decides deployment. The LLM engineer can make evidence legible but cannot approve another function's gate.

Rejection criteria should include:

1. the target residual is not stable across frozen repeats;
2. a routing, retrieval, prompt, template, schema, or judge repair removes it;
3. authorized data cannot express the distinction without leakage;
4. target evidence does not clear the preregistered criterion;
5. a protected language, citation, abstention, authorization, or consequence segment regresses;
6. resource use exceeds the approved experimental envelope;
7. artifact compatibility or reversibility is not demonstrable;
8. a required authority withholds approval.

These are stop rules, not suggestions to revisit after a preferred method wins.

Build a replay matrix that preserves cause

A frozen baseline needs more than one pass/fail column. For every target and retention case, record the input artifact, authorized evidence state, expected proposal or terminal state, actual proposal or terminal state, deterministic validator outcomes, judge identity, repeated-trial distribution where generation varies, and the first layer at which the trace diverges. Resource measurements belong beside the same run identity.

Use paired replay rows:

Row	What stays frozen	What changes	Permitted conclusion
inherited baseline	all identified dependencies	nothing	reference observation only
corrected language map	model through evaluator	one upstream mapping rule	whether the mapping explains those cases
prompt/context repair	all other dependencies	one declared message or evidence surface	whether a no-weight repair is sufficient
adaptation candidate	every non-training dependency	one versioned weight intervention	behavior difference under this experiment
confounded candidate	nothing reliable	checkpoint, corpus, and template	no causal attribution

If a managed provider cannot freeze an internal revision, the row records the exact exposed version and dated limitation. Repeated replay can reveal drift, but it cannot recover hidden identity. If an open-weight path changes a kernel or quantization while adapting, either restore the frozen runtime or preregister the combined change as a new question. Do not call two non-equivalent units paired.

The matrix also guards against outcome laundering. A candidate may improve the shorthand target while producing more unsupported citations. It may pass a mean duration budget while failing a protected tail. It may output a syntactically valid object that changes the meaning of a warranty field. Target evidence and every retained criterion remain visible; no weighted average is allowed to cancel a hard gate.

Do not learn the evaluator by accident

Freezing evaluation means preserving case-family boundaries and judgment limitations. Training examples, validation examples, and final evaluation families must not be near-duplicates. Prompt examples and retrieval documents can leak expected strings too. Record every transformation and similarity check, but do not claim that deduplication proves absence of contamination.

Automated RAG or model-based metrics can help locate a failure. They do not confer truth or authority. The accepted evaluation guidance supports datasets, criteria, and regression loops, while the RAG evidence shows that retrieval changes the model system; neither says that weight adaptation is the right repair for Mosaic. The Tülu 3 case offers a documented staged post-training recipe and evaluation practice. It is bounded research evidence from particular models, datasets, infrastructure, and benchmarks, not an operational standard, a guaranteed recipe, or evidence that Mosaic should reproduce it.

`LLME-CASE-008` is therefore used as a method case, not an outcome transplant. Its useful lesson is structural: data construction, supervised training, preference stages, and evaluation can be documented as separable phases with artifacts. Mosaic borrows the demand for traceable stages. It does not borrow dataset fitness, hyperparameters, scores, infrastructure feasibility, or a reason to perform every stage. Even a well-documented public recipe cannot answer whether Mosaic has an adaptation-sensitive residual.

Keep new evidence distinct from the inherited migration

The shorthand cases arrive after the Volume 1 migration packet. Give them a new population identifier and provenance record. Do not append them to the old migration comparison and then describe the combined set as if it had always been frozen. Do not use the new target cases to excuse the candidate provider's earlier protected regression.

The migration question asks whether a replacement adapter preserves the qualified system contract. The adaptation question asks whether a stable residual in an identified base system is plausibly sensitive to a weight intervention. They can involve the same language segment and still require separate baselines, controls, and decisions.

This separation prevents two common shortcuts. First, a migration failure cannot be rebranded as evidence that the current model needs training. Second, a successful future adapter cannot be rebranded as evidence that the held provider migration was safe. Each claim stays attached to the system unit and experiment that produced it.

Failure injection: reject the confounded candidate

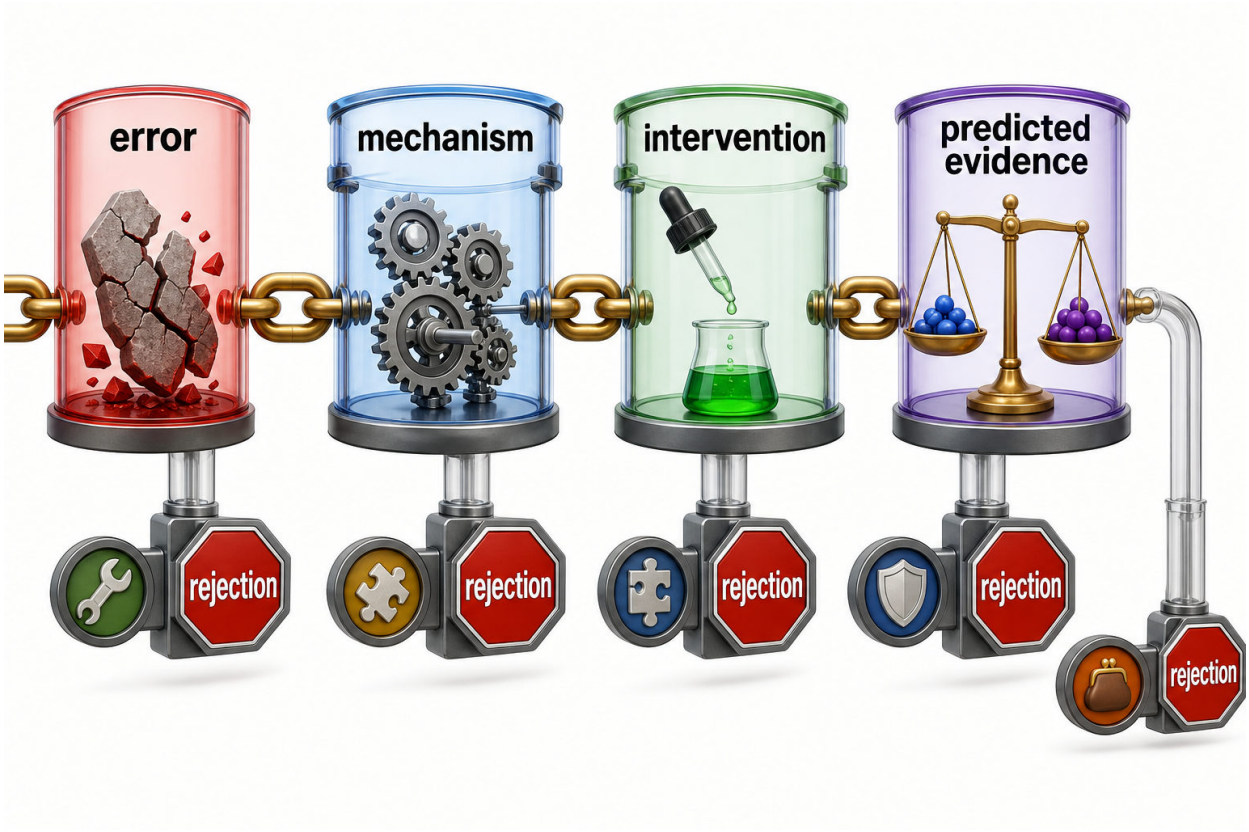
An impatient experimenter introduces `candidate-confounded-01`. It changes three surfaces at once:

- the open-weight checkpoint;

- the retrieval corpus revision;
- the serving chat template.

The candidate appears to improve two shorthand cases. No causal statement survives. The changed corpus may contain the expected phrase, the template may alter parsing, or the checkpoint may behave differently. The audit marks the result `rejected-confounded` and preserves it as a warning, not a baseline replacement.

Upstream defects must be ruled out before attributing a failure to model weights. [CLM-003] That claim does not mean every upstream cause can be exhaustively eliminated. It means the team must test plausible simpler layers and state what remains unknown before spending adaptation evidence.



V2-F01.2 - A useful hypothesis can lose. Essential labels: error, mechanism, intervention, predicted evidence, rejection. Broken-link and stop-gate shapes supplement color. The figure supports `CLM-002` and `CLM-003`; it predicts no adaptation outcome.

> **Text alternative:** A falsifiable chain connects an observed error to a proposed mechanism, intervention, predicted evidence, and explicit rejection gates.

Open MD-09 with an audit, not a run

The outgoing artifact is `mosaic-frozen-adaptation-baseline.json`. It binds the inherited digest, frozen identity, target and retention criteria, controls, budgets, authority decisions, residual ledger, hypothesis, and rejection rules. Its final disposition is intentionally narrow: `eligible-for-method-selection-not-training`.

That status does not reverse `MD-08`. The provider migration remains held. The retrieval authorization defect remains separate. No production release is approved. The new synthetic shorthand residual is a distinct investigation population, not retroactive proof that the migration regression needed adaptation.

Practice: try to disprove the adaptation story

Using the companion:

1. verify the inherited handoff path and digest;
2. reconstruct the complete provider-neutral baseline identity;
3. split malformed-language-code cases from the residual;
4. write one mechanism-specific hypothesis and predicted evidence;
5. freeze target, retention, control, budget, and rejection criteria;
6. reject the three-change candidate as confounded;
7. name every external authority and unresolved decision;
8. explain why `eligible-for-method-selection-not-training` is not training approval.

Pass when another reviewer can replay the baseline, locate each residual at a layer, disprove at least one apparent weight failure with an upstream repair, and tell exactly what evidence would stop the investigation. Fail when a checkpoint name substitutes for system identity, aggregate improvement overrides a protected gate, an evaluation case leaks into training, or tuning begins because it is fashionable.

Chapter 2 receives the frozen baseline and the unresolved open-weight candidate identity. It may inspect artifacts and compatibility. It may not train, select a method, or infer quality from architecture.

2. Inspect the Model and Tokenizer Boundary

Inspect model, tokenizer, template, precision, and runtime as separate compatibility surfaces without turning architecture into a quality forecast.

The artifact boundary is part of behavior

Chapter 1 froze a baseline and permitted method selection to be investigated, not training. The next temptation is to read a model card, count parameters, glance at an architecture diagram, and announce what should be tuned. Mosaic Desk instead performs an artifact compatibility inspection.

The fictional open-weight candidate is represented by synthetic identifiers. No named model is endorsed, and no generated digest corresponds to a real checkpoint. The exercise asks a narrower question: can the team identify the exact weights, tokenizer, template, generation defaults, precision path, runtime, and hardware assumptions that would make a later experiment interpretable and reversible?

Tokenizer, template, architecture, position/context mechanism, precision, and checkpoint are separate model-system surfaces. [CLM-004] They often arrive in one repository or service bundle, but they do not become one variable. A mismatch can change token boundaries, role markers, attention masks, padding, numerical behavior, memory use, or generated sequence shape without any intentional adaptation.

Draw the compatibility tuple

For an open-weight route, bind at least:

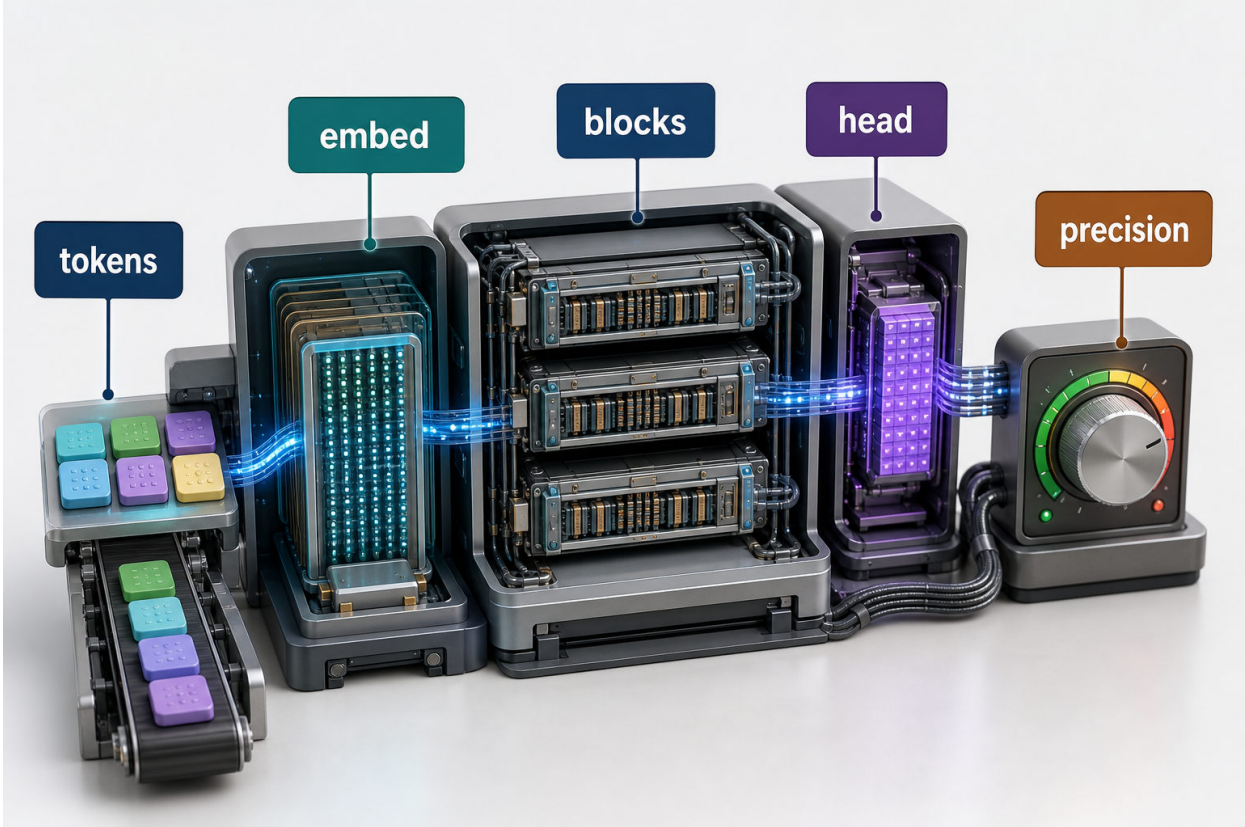
- weights artifact digest and upstream revision;
- configuration digest and architecture family;
- tokenizer vocabulary, merge/model files, normalization, added tokens, and special-token map;
- serving chat template and role-marker contract;
- context/position mechanism and configured maximums;
- dtype, quantization format, calibration artifact where relevant, and accumulation policy;
- generation defaults and explicit overrides;
- framework, runtime, attention implementation, kernels, and dependency lock;
- hardware class, device topology, memory envelope, and serving scheduler assumptions.

The tuple is an equality requirement for replay, not a claim that the parts are good. `same weights + different tokenizer` is a different system. `same repository revision + different chat template` is a different system. `same checkpoint + different quantization` is a different numerical and resource path. The exact consequences require tests.

Give the tuple one immutable manifest digest and make every later dataset, adapter, optimizer, evaluation, export, and serving record point to it. Never allow a human-friendly model name to resolve dynamically during an experiment. If a registry tag is mutable, resolve it once to an immutable revision, record the resolution time and source, and retain the allowed artifact under the applicable license and security policy.

Artifact provenance matters alongside bytes. Record publisher, source repository, license version, download route, signature or checksum where provided, local verification, modifications, and access restrictions. These records help security and legal reviewers; they do not make the LLM engineer the authority for supply-chain acceptance or usage rights.

Managed access exposes a smaller tuple: provider, endpoint, dated model identifier or immutable version where available, API contract, region or processing posture, tool/schema options, explicit generation settings, and the provider's declared lifecycle. Hidden tokenizer or runtime details become limitations. A managed comparator can still be replayed at the semantic boundary, but the team must not pretend it offers open-weight inspection.



V2-F02.1 - Inspect separate surfaces inside one model bundle. Essential labels: tokens, embed, blocks, head, precision. Texture and component shape supplement color. The figure supports `CLM-004`; it is not a performance ranking.

Text alternative: A cutaway model system shows tokens entering embeddings and transformer blocks before a prediction head, with precision recorded as a separate runtime surface.

Inspect tokens before discussing examples

Tokenization converts text into model input identifiers. Subword algorithms can represent open vocabularies, but the segmentation and length of a word or sentence depend on the learned vocabulary

and normalization rules. A domain abbreviation may be one token under one tokenizer and several pieces under another. Gujarati text, Latin transliteration, punctuation, combining marks, or code-switched strings can expand differently.

Token fragmentation can change effective sequence length and multilingual/domain representation. [CLM-005] It does not by itself prove lower semantic quality. Sequence length changes context and compute pressure; unusual fragments may indicate that examples deserve testing. The correct conclusion is to measure the target text with the exact tokenizer and then evaluate behavior.

Mosaic's tokenizer trace uses short fabricated strings rather than customer text. For each string it records Unicode form, language tag supplied by the test fixture, token pieces, token IDs, total length, special tokens, and round-trip decode. It compares tokenizer A with tokenizer B only to expose boundary differences. The counts are pedagogical fixtures, not measurements of a real model.

Check these failure surfaces:

1. normalization changes a meaningful code or combining mark;
2. added tokens exist in the tokenizer but not in the resized embedding matrix;
3. beginning/end, padding, or unknown token IDs differ from the model configuration;
4. padding direction conflicts with the generation path;
5. truncation removes an evidence or schema boundary;
6. chat-template role tokens are duplicated or omitted;
7. tokenizer revision and weights revision drift independently;
8. encode/decode round trips alter a field that must remain exact.

The injected fixture sets `pad_token_id` equal to `eos_token_id` while the proposed batching path treats padding as a normal masked region. That setting is not universally wrong, but it is incompatible with the declared serving assumption and therefore blocks the candidate until explicitly tested or corrected. The validator reports the relationship; it does not claim a quality loss.

Test special tokens as a state machine

Do not inspect special-token IDs only as a list. Exercise the paths that use them:

- a single unpadded request;
- left- and right-padded batches where supported;
- an empty or minimum-length request;
- a sequence that reaches the declared limit;
- assistant generation with and without an existing assistant prefix;
- tool or structured-response roles if the contract allows them;
- encode, serialize, generation-boundary, and decode transitions.

Record attention masks and the exact prompt tokens for these fixtures. A padding token equal to an end token can be valid under some implementations, but the mask, loss, stopping rule, and batching path must

agree. The inspection blocks only because Mosaic's declared synthetic batcher requires distinct semantics. It is a local compatibility result, not a universal tokenizer rule.

V2-F02.2 - Token boundaries change the experiment surface. Essential labels: tokenizer A, tokenizer B, length. Brackets and count blocks supplement color. The figure supports `CLM-005`; it makes no multilingual quality claim.

Text alternative: The same Gujarati-English shorthand string is divided differently by tokenizer A and tokenizer B, producing visibly different sequence lengths without asserting which is better.

Treat the chat template as an executable interface

Instruction-tuned chat models are generally trained around particular role and control-token formats. A serving template serializes system, user, assistant, and tool content into tokens. Using the wrong template can omit expected markers, duplicate generation prompts, or make trusted and untrusted fields ambiguous.

The failure injection pairs the frozen synthetic checkpoint with a template from another artifact family. The messages look reasonable before serialization, yet the emitted control-token sequence differs from the recorded training-facing contract. The inspection returns `template-incompatible`; it does not run an output and then rationalize the mismatch.

Record template source, digest, allowed roles, special tokens, generation-prompt behavior, tool/schema serialization, whitespace controls, and a golden serialized fixture. Validate the golden bytes or token IDs under the exact tokenizer. If a managed service owns serialization internally, test the public message behavior and declare the hidden boundary.

Templates also carry the Volume 1 trust rule: trusted system control and untrusted user or retrieved data remain structurally distinct. A model template cannot grant authorization, and special tokens do not make hostile text safe. External deterministic controls still authorize retrieval, validate proposals, and guard effects.

Create two golden fixtures for every allowed conversation shape. One stores the readable structured messages. The other stores the serialized text or token IDs produced by the exact template and tokenizer. Review changes at both layers. A whitespace, role, tool-call, or generation-prompt change that looks harmless in structured JSON may move token boundaries or control markers.

For adaptation, preserve the same serialization when constructing examples and serving the resulting artifact unless an isolated experiment explicitly tests the difference. Training with one role format and evaluating with another makes a poor result uninterpretable. A seemingly good result can be equally misleading if the new template embeds answers or case-specific hints.

Inspect architecture without fortune-telling

The Transformer paper establishes attention-based sequence modeling as a foundational mechanism, with results bounded to its historical tasks and configuration. `LLME-CASE-001` uses it to explain tokens, embeddings, blocks, and next-token computation. It does not establish modern chat behavior, adaptation suitability, or Mosaic quality.

Architecture inspection identifies plausible intervention surfaces but cannot predict task improvement without experiments. [CLM-006] The number of layers, hidden size, attention pattern, position mechanism, normalization, activation, tied embeddings, and output head affect compatibility and possible trainable modules. They do not tell you that LoRA rank eight will work, that a longer context will use evidence well, or that a particular language will improve.

Long-context studies show position-sensitive behavior in tested settings, not a universal rule for all current models. FlashAttention demonstrates an exact, IO-aware attention implementation with reported speed and memory results on studied hardware and workloads; it is not a quality improvement guarantee or a Mosaic capacity measurement. Record attention implementation as runtime identity because it can alter resource feasibility and numerical paths. Do not cite it as proof of task fitness.

Make precision and runtime observable

Precision is not a single label. Record storage dtype, compute dtype, accumulation behavior, quantization scheme, scale and zero-point metadata, calibration inputs where used, kernel, and device support. Mixed-precision documentation describes available mechanisms and APIs, not a universally safe configuration. Quantization may reduce memory while changing numerical behavior. Only target evaluation can bound the effect.

Run deterministic compatibility checks before any expensive evaluation:

- referenced files exist and match recorded digests;

- configuration dimensions agree with weight tensors;
- tokenizer size and embedding/output dimensions agree;
- special-token IDs resolve and satisfy declared batching behavior;
- template tokens exist and golden serialization matches;
- context and position settings are internally consistent;
- requested dtype/quantization is supported by the runtime and hardware plan;
- dependency and kernel versions are locked;
- adapter targets, if later proposed, name modules that actually exist.

These checks can prove internal consistency of a fixture. They cannot prove output quality, safety, fairness, capacity, legal permission, or release readiness.

Produce an inspection packet another engineer can challenge

The output should contain both facts and unresolved questions:

Surface	Evidence	Blocking question
weights/config	immutable revisions, digests, tensor/config agreement	are modifications and provenance accepted?
tokenizer	file digests, vocabulary size, special-token map, traces	do padding and embedding assumptions agree?
template	digest, allowed roles, golden serialization	does it match the intended artifact family?
context/position	mechanism, configured limits, test lengths	are limits supported by target evidence?
precision	storage, compute, accumulation, calibration	is the numerical path within the experiment scope?
runtime	framework, dependencies, kernels, hardware class	can the tuple be reproduced and rolled back?

For each row, name an evidence owner and a decision owner. The LLM engineer may verify a digest and demonstrate a mismatch. Security decides whether provenance is acceptable. Platform/SRE decides whether the runtime is supportable. Privacy and legal authorities decide whether local artifact or example handling is permitted. The inspection packet should make these handoffs easy rather than merging them into a generic `approved` flag.

The packet also contains a managed comparator ledger. It lists which semantic request settings and output states can be replayed and which internal surfaces are undisclosed. A future comparison may still be useful, but causal claims must stay at the observable system boundary. Do not infer a tokenizer change from output length or a quantization change from a different answer.

Separate inspection from ownership

Data owners decide whether examples may be inspected or adapted. Privacy and legal authorities decide retention and rights. Security reviews serialization and artifact supply-chain risks. Platform/SRE owns deployment infrastructure, capacity, and operational recovery. Model publishers own their licenses and disclosures. LLM engineering records compatibility and behavior evidence and raises gaps.

The synthetic artifact is marked `inspection-blocked` because its template is mismatched and pad/EOS assumption is unresolved. That is a successful inspection result. It prevents a later training run from producing evidence about an accidental system.

Practice: prove identity before asking about quality

Using `mosaic-model-tokenizer-inspection.json`:

1. verify every artifact and configuration digest;
2. compare tokenizer A and B traces without declaring a winner;
3. inspect special-token and padding assumptions;
4. reproduce the golden template serialization;
5. name architecture surfaces that could accept an intervention;
6. record precision, runtime, dependency, and hardware constraints;
7. contrast the managed comparator's exposed and hidden fields;
8. explain why the result blocks training but predicts no task outcome.

Pass when another engineer can assemble the exact intended compatibility tuple, detect both injected mismatches before evaluation, and state which surfaces remain opaque. Fail when a repository name substitutes for digests, a tokenizer count becomes a language-quality score, architecture becomes a method recommendation, or a hidden managed surface is fabricated.

Chapter 3 receives the frozen hypothesis plus an inspection with explicit blockers. It may compare intervention classes and reject alternatives. It still may not train while compatibility blockers or authority decisions remain.

3. Choose the Smallest Adaptation Ladder

Map each residual failure to the smallest plausible intervention, reject mismatched methods, and attach disconfirmation and stop rules before training.

Choose a mechanism, not a trend

The first `MD-09` artifact froze a falsifiable shorthand-mapping hypothesis. The second found that the synthetic open-weight candidate is not yet compatible: its serving template is mismatched and its padding assumption is unresolved. Chapter 3 therefore cannot approve training. It can do something more valuable first: map each residual to the smallest intervention that could plausibly change its mechanism, then reject alternatives whose evidence, access, cost, reversibility, or authority boundaries do not fit.

An adaptation ladder is not a universal ranking from cheap to impressive. It is a sequence of questions. At every rung, ask:

- what observed mechanism would this intervention change?
- what evidence would distinguish success from noise?
- what new data, access, compute, runtime, and control obligations appear?
- what retained behavior could regress?
- what result or boundary makes us stop?

No rung is earned by industry fashion. A later rung is not more mature. The best decision can be repair, replacement, or no change.

Classify the three Mosaic failures

The MD-09 ledger contains three constructed classes.

Malformed upstream language codes. These failures disappear when the stale mapper is repaired. Their mechanism sits before the model. The selected action is input-routing repair plus regression tests. They must not enter an adaptation dataset as examples of model weakness.

Stale policy knowledge. The desired answer changes when the authorized service-policy document changes. Encoding the current policy into weights would create a freshness and provenance problem. The selected action is retrieval/index/assembly repair with source-state and absence behavior, not tuning.

Stable appliance shorthand to typed fields. After routing, authorized retrieval, template, schema, and evaluator controls pass, a bounded mapping error repeats on frozen family-separated cases. This is the only current adaptation-sensitive candidate. Even here, the conclusion is merely to investigate the smallest supervised weight-changing pilot after the Chapter 2 compatibility blockers clear.

Prompt/context repair, SFT, PEFT, preference optimization, distillation, continued pretraining, and model replacement target different failure hypotheses. [CLM-007] The categories overlap in implementation, and research results depend on model, data, objective, and evaluation. `LLME-CASE-008` is the bounded staged-recipe case, not a mandatory sequence. The ladder is a diagnostic map, not a promise that one method fixes its named failure.

Rung zero: keep the frozen baseline

Every ladder begins with `no change`. The qualified behavior baseline remains the comparator and fallback. A proposal must beat it on the target criterion without failing retention or control gates. If the target is not valuable enough, evidence is too weak, rights are unresolved, or no feasible intervention is reversible, stop at the baseline.

No-change also preserves learning. A failed candidate should not silently mutate the reference. Record the candidate as rejected, retain raw transitions, and keep the baseline artifact addressable.

A baseline may be imperfect and still be the correct current decision. If no intervention clears its evidence and authority gates, retain the baseline within its already-qualified scope, preserve abstention and fallback, and narrow exposure where required. "Do nothing" never means ignore an incident; it means do not change this model surface while repairs, containment, or evidence collection occur elsewhere.

Rung one: repair task, input, prompt, context, or retrieval

These are system changes, but not weight adaptation. Clarify an ambiguous task contract; fix the language mapper; normalize an input field; correct role serialization; remove an irrelevant example; change an authorized query; repair source filtering; improve context assembly; or tighten schema validation. Each change still needs isolated evaluation.

This rung fits Mosaic's language-code and stale-policy classes. It also serves as the control for the shorthand residual: if a small prompt or context change reliably removes the error without protected regressions, the weight-change hypothesis is disconfirmed.

Retrieval is not a harmless fallback. Corpus permission, provenance, freshness, ranking, context injection, citation, privacy, and latency controls remain. Chapter 1's revoked-source incident is still separate and contained; the ladder cannot hide it inside model work.

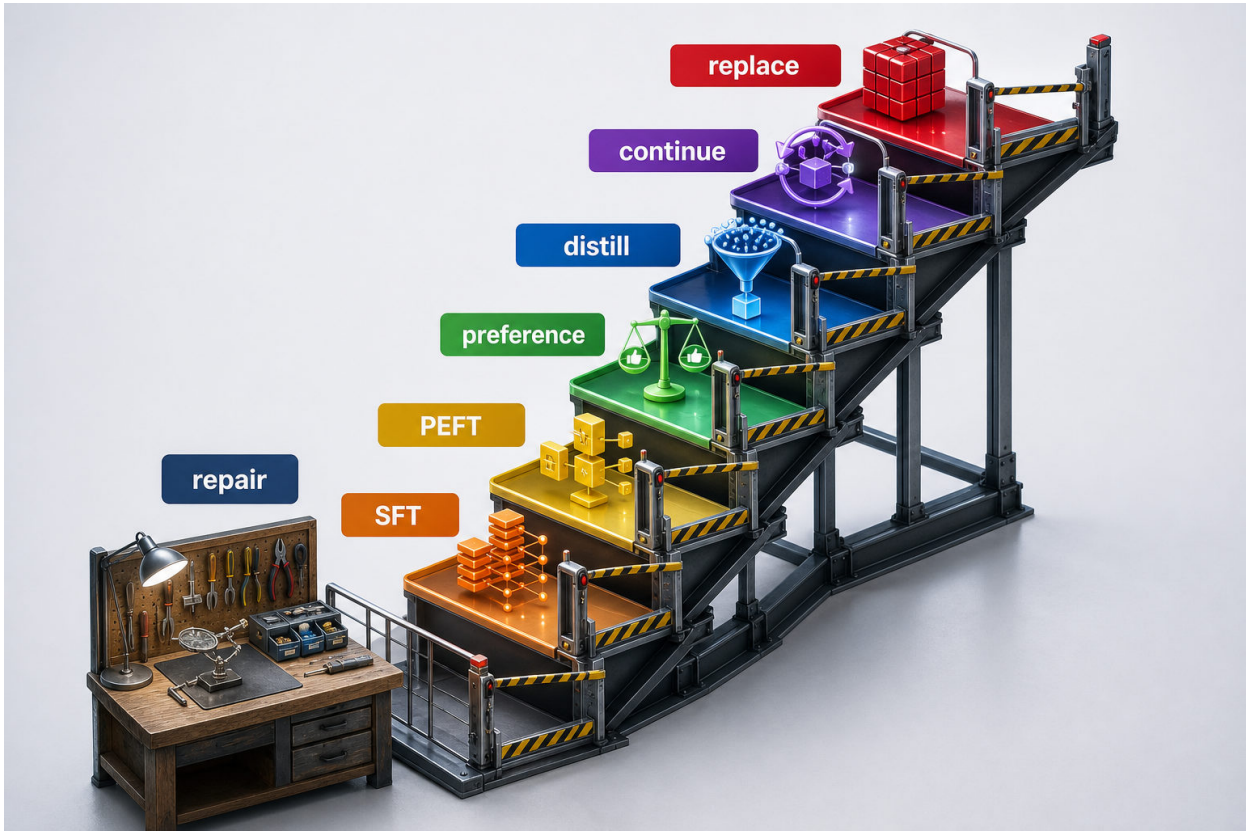
Rung two: supervised full or parameter-efficient adaptation

Supervised fine-tuning changes behavior using input-output examples under a training objective. It is plausible when the desired transformation is stable, examples demonstrate it directly, and inference-time evidence alone is insufficient. Full-parameter SFT offers broad weight access but raises compute, storage, rollback, retention, and catastrophic-regression obligations.

Parameter-efficient fine-tuning updates or adds a smaller set of parameters. LoRA introduces low-rank trainable matrices around selected weights; PEFT libraries expose multiple adapter methods; QLoRA combines low-rank adapters with a quantized frozen base in its studied setting. `LLME-CASE-009` bounds this option: the papers report reduced trainable or memory envelopes and particular experimental results, not universal equivalence to full tuning, not a guaranteed resource plan, and not evidence that Mosaic will improve.

For the shorthand candidate, a supervised PEFT pilot is smaller and more reversible than full-parameter SFT if, and only if, the target modules exist, the base/tokenizer/template/runtime tuple is fixed, the data are authorized and representative, adapters can be versioned independently, and evaluation can detect target and protected changes. The selected status is `pilot-investigation-proposed`, not `training-approved`.

Small parameter count does not automatically mean small operational risk. An adapter can be applied to the wrong base, composed in the wrong order, merged irreversibly, served with an incompatible template, or loaded without the intended authorization boundary. Record base digest, adapter digest, target modules, rank and scaling configuration, training-data recipe, optimizer and seed, checkpoint policy, merge state, load order, runtime support, and rollback procedure. Later chapters will deepen these records; method selection only proves they are required.



V2-F03.1 - Climb only when the failure mechanism demands it. Essential labels: repair, SFT, PEFT, preference, distill, continue, replace. Stop signs and distinct platform shapes supplement color. The figure supports `CLM-007` and `CLM-009`; height does not mean quality or maturity.

Text alternative: A gated intervention ladder moves from repair through supervised and parameter-efficient tuning, preference optimization, distillation, continued pretraining, and replacement, with a stop gate at every rung.

Rung three: preference optimization

Preference optimization fits cases where the training signal is a comparison between acceptable behaviors rather than a single correct target. Direct Preference Optimization provides one studied objective that avoids an explicit reward-model-and-RL loop. `LLME-CASE-010` demonstrates a bounded method and benchmark evidence. It does not establish universal human values, annotator representativeness, safety, domain correctness, or product authority.

Mosaic's current shorthand mapping has a typed expected field. Pairwise preference labels would add ambiguity without matching the mechanism. Reject preference optimization for this hypothesis. It might become relevant to a future bounded style or trade-off question, but only with a defined preference population, disagreement policy, consequence analysis, retention set, and authority.

Rung four: distillation

Distillation trains a student from a teacher's targets or distributions. It may fit a deployment problem where a qualified larger behavior must be approximated under a smaller runtime envelope. The original distillation evidence supports the general method in studied settings, not faithful transfer of every capability or control.

Mosaic has no qualified teacher demonstrating the desired shorthand behavior and no accepted compression objective. Distillation would copy unknown teacher errors and add another evaluator dependency. Reject it for the present residual.

Rung five: continued pretraining

Continued pretraining changes a model through additional unlabeled or self-supervised text, often to expose domain or task distributions. Domain- and task-adaptive pretraining research reports gains in studied settings. It does not guarantee that more domain text will teach a precise typed mapping, preserve instruction following, or satisfy rights and privacy constraints.

Mosaic's bounded shorthand transformation has paired expected outputs. Continued pretraining is less targeted, harder to attribute, and more demanding in data and compute than the current hypothesis requires. Reject it. It could be considered only for a broad, stable representation gap supported by authorized corpora and experiments that distinguish it from supervised alternatives.

Rung six: replace the model

Replacement can be smaller than adaptation when weights are inaccessible, a candidate already satisfies the contract, or the current artifact cannot meet a hard constraint. It also changes lifecycle, opacity, compatibility, behavior, resource, privacy, and rollback evidence. A managed replacement must replay the same semantic task dossier. An open-weight replacement needs a new full artifact tuple.

The inherited provider migration remains held because protected Gujarati citation and synthetic tail gates regressed. That failure is not permission to replace again by benchmark rank. Replacement remains a future comparator only after its own qualification.

Failure injection: fashion proposes tuning twice

The first injected proposal sends all four multilingual failures to PEFT. The frozen baseline audit disproves half of its premise: two cases are upstream language-code defects. The ladder rejects those training rows and routes them to application repair.

The second proposal sends stale policy knowledge to continued pretraining because "the model needs to know the domain." That mechanism would bury volatile facts in weights, weaken source attribution, and make refresh expensive. The ladder rejects it in favor of authorized retrieval repair. This does not prove retrieval will meet the task; it preserves the right experiment and explicit no-evidence behavior.

Only the recurring shorthand-to-field residual remains. Even that proposal cannot advance while Chapter 2's template and padding blockers stand. The failure injection therefore demonstrates three useful outcomes at once: repair upstream, repair retrieval, and pause a plausible adaptation until compatibility clears.

Let access and resources eliminate methods

Weight access, data quality, compute, reversibility, runtime, and control requirements constrain available interventions. [CLM-008] Add privacy, legal rights, security, and authority as independent gates rather than pretending engineering feasibility answers them.

For a managed model without adaptation access, SFT or PEFT may be unavailable even if conceptually plausible. A provider-hosted adaptation endpoint may expose training inputs and version IDs but hide optimizer, weights, or runtime. Record that evidence asymmetry. For open weights, technical access expands responsibility: artifact licenses and provenance, dataset rights, training infrastructure, optimizer state, checkpoints, adapter composition, evaluation, serving compatibility, and rollback all need owners.

Use an access-resource map:

Intervention	Required access	Data signal	Resource/control burden	Reversibility
repair	application/config	cases and traces	bounded but layer-specific	usually high
SFT	training or provider endpoint	authorized demonstrations	training, storage, full regression	model/version rollback
PEFT	compatible base and target modules	authorized demonstrations	adapter training/composition/serving	high only if base stays fixed
preference	preference objective access	calibrated comparisons	label governance and full regression	model/version rollback
distill	student training plus teacher access	teacher targets/distributions	teacher cost and fidelity tests	student replacement
continue	base training access	authorized corpus	high compute/data governance	checkpoint rollback

Intervention	Required access	Data signal	Resource/control burden	Reversibility
replace	alternate model/service	frozen evaluation evidence	full requalification	fallback if retained

V2-F03.2 - Feasibility is a set of gates, not a price tag. Essential labels: access, data, compute, control, reversibility. Icons and narrowing rails supplement color. The figure supports `CLM-008`; it reports no actual budget or approval.

Text alternative: Intervention candidates are filtered through access, data, compute, control, and reversibility gates before one bounded pilot investigation remains.

Attach predicted evidence and stop rules

At least one plausible intervention should be rejected using predicted evidence or boundary constraints before experimentation expands. [CLM-009] Mosaic rejects more than one: preference optimization mismatches the typed target, distillation lacks a qualified teacher, continued pretraining is too indirect, full SFT exceeds the smallest justified change, and replacement has unresolved protected regressions.

The proposed supervised PEFT pilot remains conditional on these stop rules:

1. stop if the template/padding compatibility blockers remain;
2. stop if data provenance, rights, privacy, family separation, or target coverage fails review;
3. stop if prompt/context repair clears the target criterion;
4. stop if the adapter cannot be bound to an exact base and runtime;
5. stop if the preregistered target gain is absent across repeats;

6. stop immediately on citation, authorization, abstention, conflict-escalation, schema, or protected-language regression;
7. stop if compute, memory, duration, or storage exceeds the experimental envelope;
8. stop if rollback cannot restore the frozen tuple;
9. stop if a named authority withholds its decision.

An experiment that stops still produces evidence. Do not lower a threshold, change the case set, switch judges, or add training examples mid-run without opening a new versioned experiment.

Complete the MD-09 decision without training

`mosaic-adaptation-ladder.json` records every candidate intervention, its mechanism fit, access, data, resource and control burden, predicted evidence, rejection reason, and disposition. It preserves three separate outcomes:

- malformed language codes: `repair-upstream`;
- stale policy knowledge: `repair-retrieval`;
- stable shorthand mapping: `conditional-peft-pilot-investigation`.

The overall status is `method-selection-complete-training-not-approved`. Compatibility, data, privacy/legal, resource, security, domain/product value, and experimental authority gates remain unresolved. The frozen Volume 1 baseline remains the comparator and fallback.

Preserve provider neutrality without erasing asymmetry

On a managed path, the smallest available weight-changing intervention may be a provider adaptation job with limited controls. The record must name the exposed base version, uploaded data identity, provider job configuration, returned model identifier, retention and processing terms, evaluation evidence, lifecycle, and rollback. Hidden optimizer or artifact internals remain limitations.

On an open-weight path, PEFT may expose much more experimental control but also more responsibility for data pipelines, compute, checkpoints, adapter/base compatibility, serving, and recovery. The application-level target, terminal states, protected cases, authorization, and evaluation contract remain shared across both routes. Do not score one route as inherently superior; eliminate or retain it using the frozen requirements.

Practice: defend every rejection

Use the three MD-09 companion artifacts to:

1. trace each failure class to its most plausible layer;
2. run the no-tune controls before a weight-changing proposal;

3. compare all seven intervention classes plus no change;
4. reject at least one technically feasible method because its mechanism does not fit;
5. reject at least one method because access, data, resource, or reversibility constraints do not clear;
6. write predicted evidence and disconfirmation for the proposed pilot;
7. preserve target, retention, abstention, authorization, and authority gates;
8. show that no adapter or training run was created.

Pass when another reviewer can tell why each class received a different action, reproduce every elimination, and stop the pilot before training when any prerequisite fails. Fail when cost alone selects a method, PEFT is treated as universally equivalent, preferences become values by assertion, stale knowledge enters weights by default, or `pilot` is used as a euphemism for unapproved training.

`MD-09` is now complete as a decision dossier. The next chapter may specify a data recipe for the conditional supervised candidate. It must inherit the frozen split, rights and retention unknowns, protected cases, compatibility blockers, and no-training status. A data plan can reject the candidate; it cannot retroactively make the method inevitable.

Part II - Engineer the Learning Data

Learning data are behavior instructions encoded through selection, formatting, comparison, and exclusion. A file of plausible examples can still hide unauthorized sources, duplicate families across splits, target leakage, template mismatch, rater shortcuts, omitted languages, or no cases for behavior the adaptation must retain.

Part II makes the data evidence inspectable. Chapter 4 specifies the example unit, population, sources, rights, purpose, quality, exclusions, privacy, retention, deletion, version, and acceptance gates before transformation. Chapter 5 constructs an auditable curation pipeline and freezes train, development, evaluation, retention, and control partitions with overlap reports. Chapter 6 renders instruction and demonstration records with explicit evidence links, loss regions, negative cases, and coverage limits. Chapter 7 builds bounded comparative evidence while exposing disagreement, order effects, verbosity preference, rater coupling, and privacy limits. Chapter 8 freezes the target, retention, multilingual, abstention, safety, and control lanes that no aggregate gain may erase.

Mosaic Desk advances through `MD-10` and `MD-11`: an authority-aware recipe, curation and split manifests, instruction records, a tiny synthetic preference fixture, and a protected retention/control suite. The data mechanics become reviewable, but the inherited template mismatch still blocks training readiness. No dataset size or clean split creates product value, data rights, or method fitness by itself.

The handoff into Part III is a versioned evidence package with visible limitations and forbidden regressions. The next part must freeze the training question and state, distinguish optimization behavior from held-out behavior, compare methods on the same protected contract, and retain rejection as a valid outcome.

4. Specify the Data Recipe

Turn a conditional adaptation hypothesis into a reproducible, authority-aware data recipe before collecting or transforming examples.

A folder of examples is not a dataset decision

`MD-09` ended with a conditional proposal, not a training authorization. Mosaic Desk has one recurring synthetic failure class: an identified open-weight baseline maps a bounded set of appliance shorthand incorrectly into typed fields after simpler routing, retrieval, template, schema, and evaluator repairs. The proposed next investigation is a small supervised PEFT pilot. Its template and padding compatibility blockers, data rights and retention, resource envelope, product/domain value, security review, and experimental authority remain unresolved.

Chapter 4 asks whether data capable of testing that hypothesis can be specified without borrowing customer records, contaminating evaluation, or erasing protected behavior. The answer begins with a recipe: a versioned plan that says what each example means, where it may come from, which transformations are allowed, how mixture choices connect to the behavior contract, what must be excluded, and who can authorize each decision.

The companion uses fabricated records and declarations. It reads no production export, contacts no provider, and starts no training. `authorized` is a fixture field used to exercise the gate, not a legal conclusion.

Write the behavior claim at the top of the card

A data recipe starts from the falsifiable hypothesis, target cases, and forbidden regressions. Mosaic's target is not general domain fluency. It is the shorthand-to-typed-field mapping for an authorized, bounded appliance vocabulary within structured Gujarati-English service messages. The expected output remains a proposal. Warranty decisions, customer contact, ordering, scheduling, and record mutation remain prohibited.

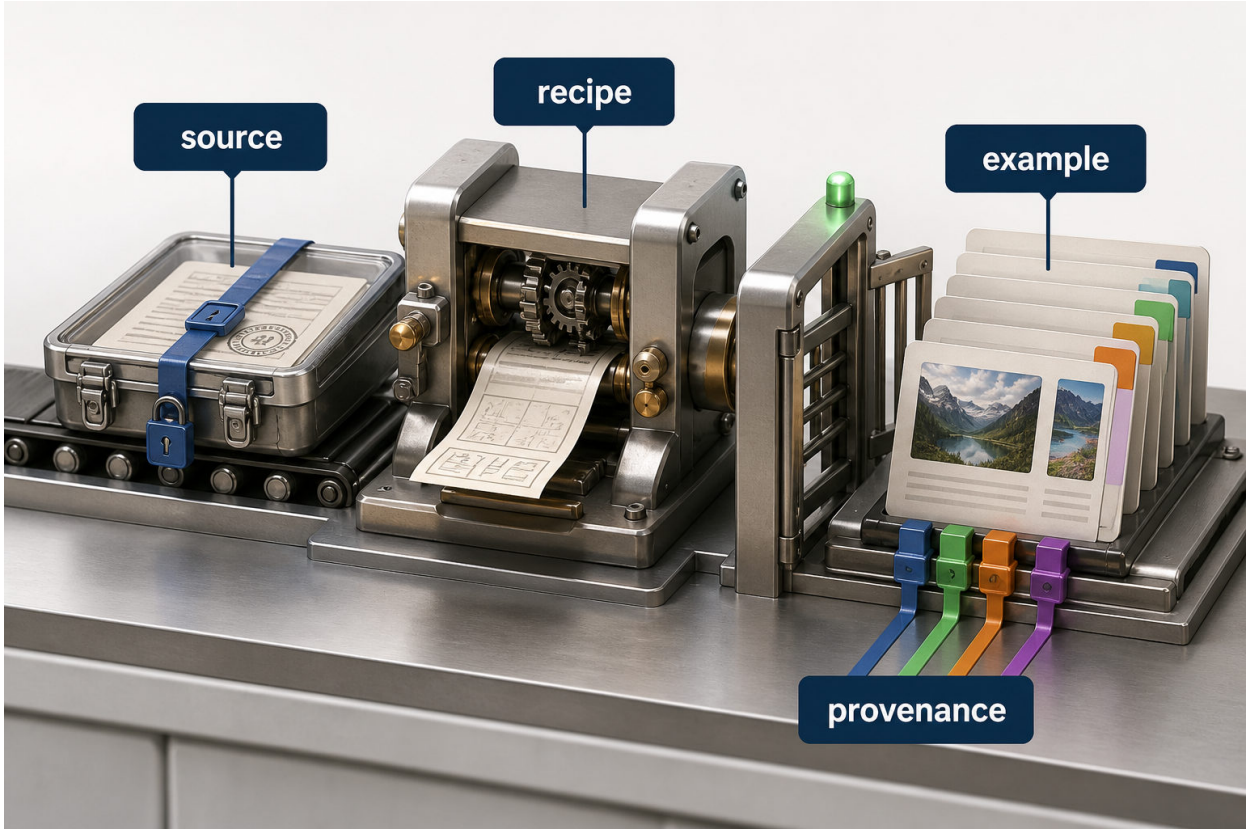
Protected clauses include correct citation, source authorization, abstention when evidence is absent, escalation on conflict, schema validity, protected-language behavior, and no external effect. Every proposed record must map to either the target or a named protected clause. An example that is merely interesting has no place in the recipe.

A data recipe identifies sources, rights/permissions, example schema, inclusion/exclusion, quality, mixture, transformations, reviewers, versions, and limitations. [CLM-010] The field list is an engineering synthesis from dataset documentation and post-training practice. Filling it out improves auditability; it does not prove lawful use, consent, representativeness, safety, or fitness.

Give the recipe an immutable identity derived from its canonical fields. Bind it to:

- the exact `MD-09` baseline and hypothesis identifiers;
- the open-weight model/tokenizer/template/runtime tuple, even while compatibility is blocked;
- behavior-contract and error-taxonomy versions;
- permitted source inventory and authority decisions;
- record schema and transformation graph;
- target mixture and sampling rationale;
- exclusions, quarantine rules, and retention/deletion states;
- split and contamination policy;
- author, reviewer, domain, privacy/legal, security, and data-owner gates;
- known gaps and stop conditions.

A recipe change creates a new identity. Do not overwrite version 1 after seeing a candidate result.



V2-F04.1 - A recipe turns sources into traceable examples. Essential labels: source, recipe, example, provenance. Shape and tab patterns supplement color. The figure supports `CLM-010`; documentation does not grant data rights.

Text alternative: Authorized source trays pass through a versioned recipe workbench into example records that retain source and transformation provenance tabs.

Define one example unit before counting examples

Mosaic's unit is a task episode, not an isolated answer string. Each raw candidate record needs:

- stable record and source-family IDs;
- source type, owner, origin, creation route, and authorization state;
- allowed purpose, access scope, retention state, and deletion trigger;
- language, script, domain, appliance class, consequence, and error slice;
- trusted task/control fields separated from untrusted message or evidence data;
- authorized context references and source revisions;
- target terminal state and typed field expectation;
- target/protected behavior clause;
- synthetic/human/transformed origin and generator configuration where applicable;
- transformation history and parent IDs;
- author, reviewer, and domain-review states;
- split eligibility, family/group key, and prohibited destinations.

Do not put the final training serialization into the raw unit. Preserve structured fields so Chapter 6 can render the exact approved chat template and loss region. This makes a later template correction reproducible without pretending the underlying example changed.

One email thread can yield multiple related records, but they share a family key. A translated, paraphrased, redacted, or synthetically expanded child retains its parent. These relationships will protect the split firewall in Chapter 5.

Inventory sources by authority, not convenience

The recipe considers only three fictional source classes.

Hand-authored synthetic scenarios. Domain-shaped but fabricated messages written from the approved vocabulary and contract. They contain no real person or customer event. A domain reviewer must still confirm the typed target and evidence relation.

Authorized policy snippets. Fabricated, versioned source passages designed solely to test citations, absence, conflict, and stale-state behavior. They remain external evidence, not facts to memorize in weights.

Generated synthetic candidates. A managed or open-weight generator may propose variants if its use, input, output handling, and retention are approved and fully disclosed. The generator's output is untrusted candidate data until independent rules and reviewers accept it.

The failure injection offers a fourth source: `production-export-convenient.csv`. Its declaration lacks an approving owner, purpose authorization, and deletion state; sampled rows contain personal data; its source families overlap frozen evaluation cases. The gate returns `reject-unauthorized-overlapping-personal-data`. Redaction after ingestion cannot retroactively authorize collection or repair evaluation contamination.

Legal/privacy/data/domain authorities approve sources and intended uses within their mandates. Security reviews handling and access controls. The LLM engineer records and enforces decisions in the data system but cannot create consent, rights, or domain truth by adding metadata.

Treat synthetic generation as a fallible source

Synthetic instruction data can scale examples but may reproduce generator errors and biases. [CLM-011] `LLME-CASE-006` bounds the documented corpus-pipeline pattern. `LLME-CASE-007` and the Self-Instruct evidence show bounded multilingual and synthetic-data methods under their studied generators, filtering, languages, and evaluations. They do not make synthetic output independent truth or establish Mosaic quality.

For every generation batch, record:

- generator/provider and exposed model identity;
- prompt/template and sampling configuration;
- seed where meaningful and request/batch identity;
- source material supplied to the generator;
- prohibited fields and pre-generation filters;
- raw candidate count, acceptance count, and rejection reasons;
- exact automated checks and their limitations;
- independent author/reviewer identities;
- language and slice distributions before and after review;
- retention and deletion status for raw generations.

Do not allow a generator to grade its own correctness without an independent criterion. Agreement with its own phrasing is coupling, not corroboration. A polished synthetic target can still invent an appliance property absent from authorized context. Chapter 6 will quarantine exactly that failure.

`LLME-CASE-008` provides a staged post-training documentation pattern. Its public data mixtures and reported outcomes are specific to its model family, sources, infrastructure, and evaluations. Mosaic copies only the discipline of versioned stages and artifacts.

Specify transformations as named functions

Write transformations before processing records. Each step has an input schema, output schema, version, allowed fields, reject states, and audit fields.

Mosaic's proposed graph is:

1. validate source authorization and allowed purpose;
2. reject or quarantine prohibited personal or secret fields;

3. preserve the immutable raw digest under approved retention;
4. normalize Unicode and structural whitespace without changing appliance codes;
5. segment thread turns while retaining boundaries and parent identity;
6. validate language/script labels rather than trusting upstream codes;
7. map approved shorthand vocabulary to a target annotation candidate;
8. attach evidence and behavior-clause references;
9. run exact and approximate family/holdout overlap checks;
10. route to author and domain review;
11. declare split eligibility, never final split assignment;
12. publish the transformed digest and decision ledger.

Normalization must have invariants. Original text remains addressable where retention permits. Structured codes round-trip exactly. Unicode normalization is recorded. Removed headers, signatures, and quoted history are represented in the transform log. A filter may reject a row; it may not silently make it disappear.

Plan the mixture as a hypothesis

The target population is not directly observed in this fictional project. A mixture is therefore a stated experimental allocation, not a frequency estimate. Mosaic plans explicit trays for:

- Gujarati, English, and Gujarati-English structured messages;
- each approved appliance and shorthand family;
- direct positive mappings and near-miss negative mappings;
- missing, stale, conflicting, and unauthorized evidence;
- success, correct abstention, and escalation states;
- short and long structured inputs within the approved context budget;
- target shorthand failures and every protected retention clause.

Multilingual mixtures need explicit language/task balance and separate evaluation. [CLM-012] SentencePiece and Aya evidence establish that tokenization and multilingual coverage are material design dimensions in studied systems. They do not define an optimal Mosaic mixture or prove equal quality across languages.

Measure every proposed slice using the exact tokenizer from Chapter 2. Record token lengths, truncation risk, and fragmentation distributions without turning them into quality scores. If Gujarati examples consume more tokens, the remedy is not automatically to downsample them. Revisit the context, template, batching, and resource hypothesis while protecting the behavior requirement.

Make allocation rules executable

For each tray, state an allocation rule and a shortfall behavior. A rule can request a minimum number of distinct source families, not merely rows. It can cap children from one generator prompt, require independent review for every high-consequence target, and reserve explicit proportions for non-answer states. If eligible families cannot satisfy the rule, the recipe reports a gap; it does not copy, translate, or paraphrase records until the counter turns green.

Record the denominator behind every mixture view. `40% Gujarati` could mean rows, unique families, input tokens, target tokens, or reviewed cases. These have different implications. Report at least records and unique families, then use token counts for resource planning. Never describe the synthetic allocation as a user-population prevalence.

Mixture changes are experimental variables. If the team later increases abstention examples, it creates a new recipe and a new hypothesis about behavior. It must not compare the resulting candidate with the old baseline as though only adapter configuration changed.

V2-F04.2 - Planned balance makes convenience gaps visible. Essential labels: language, domain, error, abstention, gap. Patterns and empty tray outlines supplement color. The figure supports `CLM-011` and `CLM-012`; it reports no real population frequencies.

Text alternative: A target-population outline is compared with separate language, domain, error, and abstention trays, leaving convenience-sampling gaps visibly empty.

Freeze exclusions and known gaps

Exclude frozen evaluation, retention, and control cases plus every derivative family. Exclude sources without explicit purpose authorization. Exclude unresolved personal data, secrets, unsafe handling,

unverifiable targets, incompatible template records, unknown parentage, unsupported languages, and examples whose desired output would cross Mosaic's authority boundary.

Quarantine differs from rejection. Quarantine retains a bounded record for an approved reviewer to resolve an ambiguity. Rejection states that the row cannot enter the recipe version. Deletion follows the governing authority's retention rule; it is not an engineer's informal cleanup.

Retention validity must be evaluated at use time, not only source-registration time. An approval can expire, a purpose can change, a deletion request can arrive, or an upstream source can be revoked. The recipe records how a source owner notifies the pipeline, which derived records and manifests are affected, whether deletion is required, and what happens to any trained artifact. This chapter creates no trained artifact, so the last branch remains an unresolved future control rather than a fictional unlearning promise.

Keep raw, transformed, rejected, and aggregate audit retention separate. A short-lived generated candidate may be deleted after review while a non-content reason code and batch count remain. Conversely, a reproducibility need cannot override an authority's deletion instruction. Where reproducibility and deletion conflict, record the limitation and stop the experiment if necessary.

Known gaps remain first-class: the recipe has no real target-population prevalence, no authorization for customer content, no evidence that synthetic language reflects regional use, no approved data-retention period, and unresolved model/template compatibility. A complete data card can still end with `recipe-specified-data-gate-pending`.

Complete the first half of MD-10

`mosaic-data-recipe.json` records the hypothesis, example schema, source registry, authority matrix, transformations, mixture plan, exclusions, split firewall, gaps, and stop conditions. The convenient export is rejected. No raw customer data is copied, and no record enters training.

The recipe advances only to `recipe-specified-data-gate-pending`. Chapter 5 must execute it against synthetic fixtures, preserve rejected rows, test false duplicate removal and holdout leakage, and decide whether clean separated partitions can be frozen. If they cannot, the conditional adaptation project stops.

Managed generation does not become a second truth source

A managed model may help propose synthetic variants while the open-weight candidate is the intended training target. Keep the routes distinct. Record what data leave the application boundary, the provider's exposed model/version, processing and retention terms approved by competent authorities, prompt/template identity, and every returned candidate. The managed service cannot see frozen evaluation targets and cannot approve its own outputs.

If those constraints cannot be met, omit managed generation. Hand-authored synthetic fixtures or a smaller dataset may preserve a more interpretable experiment. The purpose of the recipe is to make that tradeoff visible, not to maximize row count.

Practice: reject the easy dataset

Using the Chapter 4 companion:

1. trace the recipe to the MD-09 hypothesis and protected criteria;
2. reproduce the example schema and transformation graph;
3. evaluate every source's purpose, authority, retention, and split eligibility;
4. reject the convenient production export with all reasons intact;
5. map each mixture tray to a behavior clause and separate evaluation;
6. label synthetic origin and generator limitations;
7. list known gaps and unresolved authority decisions;
8. explain why a documented recipe is not a training approval.

Pass when another engineer can reproduce the planned record shape, sources, transformations, mixture, exclusions, review states, and recipe identity without seeing a hidden folder. Fail when documentation is treated as rights, synthetic output as truth, language balance as one aggregate, holdouts as useful training examples, or redaction as retroactive authorization.

5. Curate, Deduplicate, and Separate

> Execute the data recipe as an auditable pipeline that preserves rejected records, detects indirect overlap, and freezes purpose-separated dataset vaults.

Curation is a claim-changing operation

Chapter 4 froze `mosaic-data-recipe.json` with an explicit negative decision: a convenient production export is unauthorized, contains personal data, and overlaps holdout families. Chapter 5 does not search for replacement data. It executes the accepted recipe against a small fabricated inventory and asks whether the resulting partitions can support an interpretable experiment.

Every normalization, filter, duplicate decision, and split assignment changes what the future training result could mean. The pipeline must therefore make accepted, rejected, quarantined, and unresolved rows equally traceable. A clean output directory without a decision ledger is not clean evidence.

The companion contains synthetic strings and hashes. Similarity scores and thresholds are chosen to demonstrate decisions; they are not recommended production values, privacy metrics, or contamination guarantees.

Preserve the raw identity before transformation

Canonicalize only the record envelope needed to compute an immutable source digest. Keep content and metadata roles distinct so an ordering change in JSON does not invent a new example while a content change does. Bind every child transformation to:

- raw record digest;
- source-family and parent IDs;
- recipe and transform versions;
- authoring or generation batch;
- authority and retention state;
- exact before/after fields;
- decision, reason code, and reviewer;
- timestamps or batch sequence where required by the audit.

Do not retain raw content beyond an approved rule merely because hashing is useful. If an authority requires deletion, retain only the permissible deletion evidence and derived identifiers. A digest can remain personal or linkable data; hashing is not anonymization.

The fabricated pipeline uses content-addressed IDs to support replay. It explicitly refuses to claim that its records are anonymous, private, or licensed.

Normalize without erasing the task

Normalization should remove representational noise only when the behavior contract permits it. Mosaic records Unicode normalization form, line-ending and whitespace rules, header handling, quoted-thread segmentation, and approved signature detection. Appliance codes, evidence identifiers, language switches, message roles, punctuation used by the structured parser, and source revisions must survive.

Run invariants after each step:

1. authorized fields remain authorized;
2. task-bearing codes round-trip;
3. message and evidence boundaries remain visible;
4. language/script labels are not silently replaced;
5. source-family lineage remains connected;
6. target/protected clauses remain mapped;
7. rejected content cannot re-enter through a derivative.

If a normalization rule makes two semantically distinct records identical, the rule is too destructive for this task. Version and revise it rather than manually restoring selected examples.

Separate quality, authority, and duplicate decisions

A record can be well written and unauthorized. It can be authorized and malformed. It can be high quality and duplicate a frozen evaluation family. Use independent decision fields.

Mosaic's filter stages are:

- source authority and allowed-purpose gate;
- prohibited content and retention gate;
- structural/schema validation;
- evidence-link and target-state validation;
- language/domain/slice validation;
- exact duplicate detection;
- family and approximate overlap detection;
- split eligibility and group assignment;
- post-split contamination audit.

Each rejection has a stable reason such as `unauthorized-purpose`, `personal-data-unresolved`, `schema-invalid`, `unsupported-target`, `exact-duplicate`, or `holdout-family-overlap`. A

quarantine has an owner and next decision. Silent dropping makes acceptance rates, mixture changes, and bias impossible to audit.

Filtering and normalization choices alter the learned distribution and must be recorded as data decisions. [CLM-015] A language detector, quality heuristic, or signature remover can change which dialects, short messages, or refusal cases remain. The effect on behavior requires target experiments; a higher filter score is not a better dataset by definition.

Detect exact duplicates first

Exact duplication is representation-dependent. Compute at least a raw digest and a declared normalized-content digest. For structured episodes, consider per-field and whole-record identities. Two records with different source IDs but identical authorized input, context, and target may be duplicate learning units. Two records with identical text but different source revisions may need separate evidence treatment rather than automatic merging.

Choose a canonical survivor using a versioned rule: stronger provenance, clearer authorization, primary origin, earlier approved version, or higher reviewed completeness. Record all aliases. Never delete the duplicate relationship because a row was removed from training.

The Dolma and deduplication studies in `LLME-CASE-006` demonstrate large-scale documented filtering and duplication work in their own pretraining setting. Their thresholds, corpus rights, model results, and scale do not transfer to Mosaic's post-training dataset.

Treat near duplication as a reviewable hypothesis

Deduplication can reduce memorization and evaluation contamination but thresholds and scope create tradeoffs. [CLM-013] Approximate methods may compare character n-grams, tokens, normalized fields, embeddings, templates, metadata, or known transformation graphs. Each representation misses some overlap and invents some similarity.

Mosaic deliberately creates two traps.

False duplicate removal. Two unrelated service messages end with the same long templated signature. A naive whole-message similarity rule marks them duplicates. The repaired pipeline removes the known signature only for similarity comparison, preserves original content under its retention rule, and compares task-bearing fields. The records remain distinct.

Indirect evaluation leakage. A training candidate paraphrases a frozen evaluation message and changes whitespace, so exact hashes differ. Both records retain the same synthetic source-family ID, and a family-aware plus approximate-content audit detects the crossing. The training candidate is rejected as `holdout-family-overlap`.

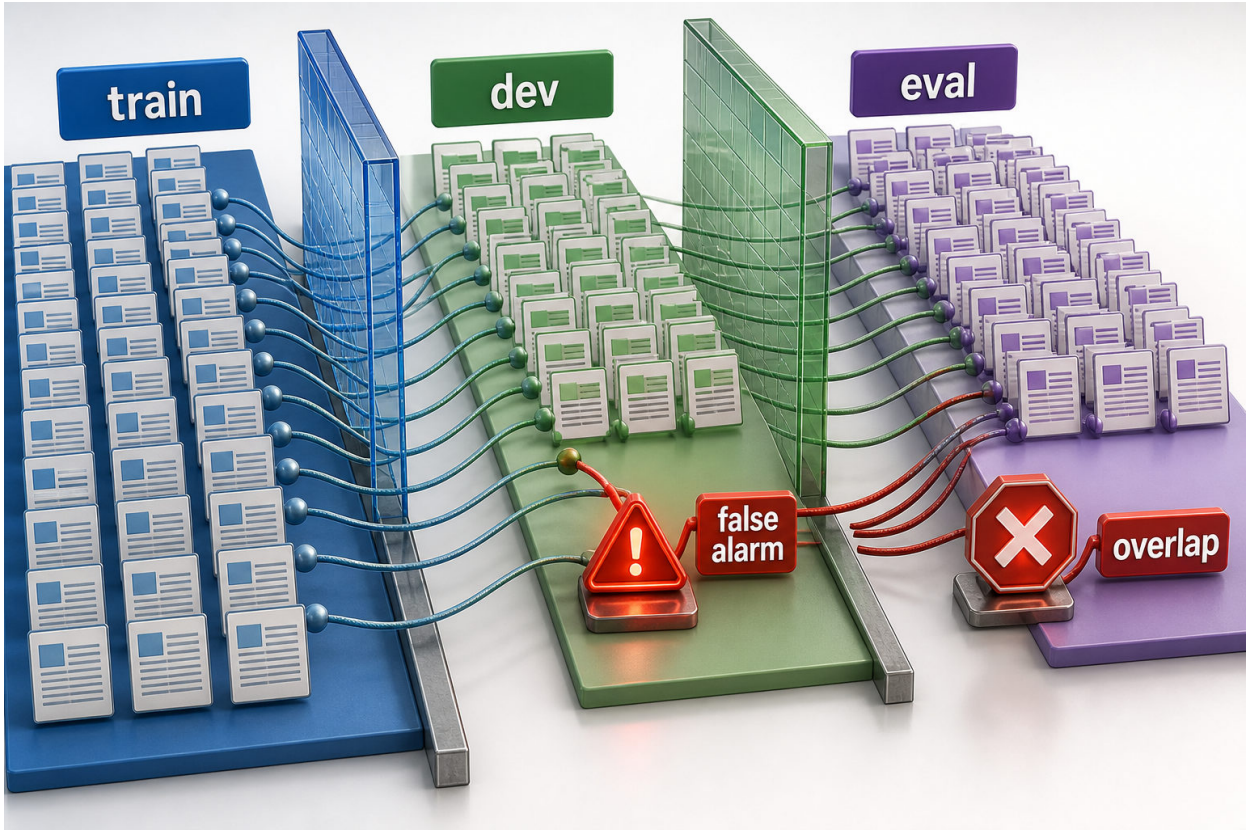
No threshold eliminates semantic uncertainty. Sample matches above and below the boundary, stratify by language and record type, record false-removal and missed-overlap examples, and keep residual uncertainty in the dataset card.

Calibrate a threshold without calling it truth

Build a small reviewed calibration set of known same-family transformations, known distinct records that share templates, and ambiguous pairs. The set itself must be versioned and separated from final evaluation. Compare candidate representations and thresholds against the reviewed relationship labels. Report false merges, missed relationships, and uncertain pairs by language and record type.

Select a threshold for a declared use. A conservative train-evaluation firewall may route ambiguous pairs to rejection or review, while within-train deduplication may retain ambiguous repetition to avoid erasing legitimate variants. One number need not govern every boundary.

Recalibrate when normalization, tokenizer, embedding model, signature rules, language mix, or source type changes. Even then, the result estimates behavior on the calibration pairs; it does not establish a universal semantic-duplicate boundary.



V2-F05.1 - Indirect duplicates threaten split claims. Essential labels: train, dev, eval, overlap, false alarm. Line patterns and alarm shapes supplement color. The figure supports `CLM-013`; it does not prescribe a universal similarity threshold.

Text alternative: Near-identical record families approach train, development, and evaluation barriers; shared signatures trigger a false alarm while a paraphrased evaluation relative triggers a real overlap alarm.

Assign groups before assigning rows

Random row splitting is invalid when records share a thread, source document, appliance incident, template seed, translation, paraphrase, generator prompt, or synthetic parent. Construct a group key that captures every known dependency, then assign the group to one purpose.

Train, development, evaluation, retention, and control sets need explicit separation and identity. [CLM-014]

- **Train** supplies supervised learning records after all gates.
- **Development** supports bounded iteration and method choices; repeated use can overfit decisions.
- **Evaluation** measures the frozen target claim and remains inaccessible to data authors and training procedures except through approved evaluation execution.
- **Retention** protects existing capabilities and forbidden-regression segments.
- **Control** diagnoses no-tune, prompt/context, routing, retrieval, evaluator, and deterministic behavior without becoming a training reservoir.

Historical overlap with a base model's pretraining may be unknowable, especially for managed services or opaque corpora. Record that limitation. The local firewall can prove only what it tested across known artifacts.

Physical and logical separation should agree. Give evaluation and retention vaults distinct access roles, storage prefixes, encryption/key policies where competent owners require them, and pipeline credentials. The training authoring process receives only train IDs and aggregate coverage gaps. A path check should fail if it attempts to resolve protected target content.

This is an engineering control design, not a security certification. Security and privacy owners decide the actual storage, access, monitoring, and retention implementation. The companion demonstrates only an allowlisted reference boundary between synthetic JSON objects.

V2-F05.2 - Purpose-separated vaults preserve claim validity. Essential labels: train, dev, eval, retention, control. Distinct vault shapes and hash seals supplement color. The figure supports `CLM-014` and `CLM-015`; separation does not prove absence of unknown historical overlap.

Text alternative: Five physically separated hashed vaults hold train, development, evaluation, retention, and control records, with family links forbidden from crossing vault walls.

Audit every boundary after the split

Pre-split deduplication is not enough. Compare train against development, evaluation, retention, and control; development against every protected set; and all derived families against their parents. Run exact, normalized, family, template-seed, and declared approximate checks. Include prompt examples, retrieval fixtures, manuscript examples, and evaluator references where they could leak expected targets.

For each pair, publish:

- comparison representations and versions;
- thresholds and why they were chosen;
- match counts and sampled decisions;
- accepted aliases, rejected crossings, and quarantines;
- language/domain/slice distribution of matches;
- known blind spots and inaccessible corpora;
- reviewer and disposition.

After any repair, recompute manifests and rerun every boundary. A moved row changes both vault identities. Do not patch a report while leaving stale hashes.

Freeze manifests and the rejected ledger

Each vault manifest contains recipe ID, curation pipeline ID, split rule, group assignments, ordered record IDs and digests, source/slice counts, transform and filter summaries, and creation state. The combined dataset card links all five identities, overlap reports, rejection ledger, known gaps, authorities, and prohibited uses.

Audit artifacts expose, not eliminate, data limitations. `LLME-CASE-008` shows the value of preserving stage-level recipe artifacts in one bounded post-training project. It cannot prove Mosaic's local data are adequate or clean. A perfect manifest can faithfully document an unrepresentative dataset.

The rejected ledger stores minimal permissible evidence: derived record ID, source family, decision stage, reason code, rule version, reviewer state, deletion status, and links allowed by retention policy. It does not become a shadow training dataset. Rejected personal content must not persist merely for debugging.

Reconcile counts as conservation of evidence

At each stage, publish a count equation:

```
input = accepted + rejected + quarantined + pending + deleted-under-authority
```

Then break it down by source, language, behavior clause, and family. A row can change state, but the transition must have a reason and version. If the totals do not reconcile, stop the freeze. Missing rows may indicate a pipeline bug; duplicated rows may indicate a retry or join error.

Counts alone can still mislead. Ten paraphrases from one parent are one family for leakage analysis. A thousand easy positive records cannot erase the absence of conflict or abstention families. Report both row and family counts plus the reviewed mixture change from recipe to curated output.

The dataset card should answer: what was intended, what entered, what changed, what was rejected, what remains, who reviewed each decision, which sets are protected, which overlaps were checked, which could not be checked, and which future events invalidate the version. It is a map of evidence and limitations, not a badge.

Complete MD-10 at a bounded gate

`mosaic-curation-manifest.json` executes the recipe on fabricated records. It preserves the signature false-positive repair, rejects the paraphrased evaluation relative, assigns known families to one of five purpose vaults, records all hashes, and reports residual unknown overlap.

The gate returns `md10-data-gate-complete-synthetic-only`. That means another engineer can reproduce the synthetic partitions and their decisions. It does not authorize real data, approve retention, clear Chapter 2 compatibility blockers, or start training.

Chapter 6 receives only the train-eligible synthetic records plus read-only coverage summaries for protected vaults. It may author evidence-linked instruction records and render template fixtures. It must not read evaluation targets into training or convert rejected rows into demonstrations.

Practice: repair a contaminated split

Using the Chapter 5 companion:

1. reproduce raw and normalized digests;
2. verify every transform and filter decision;
3. inspect the signature-driven false duplicate and retain both valid records;
4. detect the paraphrased holdout relative through family and approximate evidence;
5. assign whole groups to purpose-specific vaults;
6. rerun every cross-vault audit after repair;
7. reconcile record counts with accepted, rejected, quarantined, and deleted states;
8. state what the final manifests cannot prove.

Pass when hashes, transformations, rejection reasons, group rules, split identities, overlaps, and residual uncertainty reproduce. Fail when a universal similarity threshold is claimed, deduplication is treated as privacy, rejected rows disappear, evaluation relatives cross into training, or manifest completeness becomes a data-fitness claim.

6. Build Instruction and Demonstration Data

Turn clean synthetic partitions into evidence-linked, template-correct instruction records with intentional loss regions and visible coverage gaps.

Polished answers are not learning records

`MD-10` produced reproducible synthetic partitions. It did not produce text that should be fed directly to a trainer. Chapter 6 opens `MD-11` by converting train-eligible episodes into instruction and demonstration records whose behavioral purpose, evidence, rendering, and review are inspectable.

The managed baseline remains evaluation-only. The conditional open-weight candidate remains blocked on template and padding compatibility from Chapter 2. No trainer runs in this chapter. Rendering a record proves only that the data interface is internally consistent under a synthetic template fixture.

The goal is not to collect beautiful responses. It is to encode a diagnosed distinction: given an authorized task, structured input, and evidence state, which typed proposal or terminal state should the model be trained to produce, and which tokens should contribute to the supervised loss?

Map every example to a behavior clause

Instruction examples must map to target behavior clauses, errors, and segments rather than convenience prompts. [CLM-016] The claim is a task-specific synthesis from instruction-tuning and multilingual post-training evidence. Coverage improves the basis for an experiment; examples do not guarantee generalization or dependable behavior.

Mosaic uses four primary record objectives:

1. **positive mapping:** expand an approved shorthand into the correct typed field when authorized evidence supports it;
2. **negative distinction:** avoid a tempting but wrong mapping between similar appliance codes;
3. **abstention:** emit the explicit abstention state when required evidence is missing or stale;
4. **escalation:** preserve conflict and authority boundaries when sources disagree or a decision exceeds the proposal scope.

Each objective is crossed with Gujarati, English, and Gujarati-English slices; appliance families; short and long structured inputs; citation states; and the target/error taxonomy. The recipe never asks a language model to learn current policy facts that belong in retrieval.

`LLME-CASE-007` makes multilingual coverage an explicit data and evaluation decision in one research project. It does not establish Mosaic's language adequacy, translate regional authority, or justify

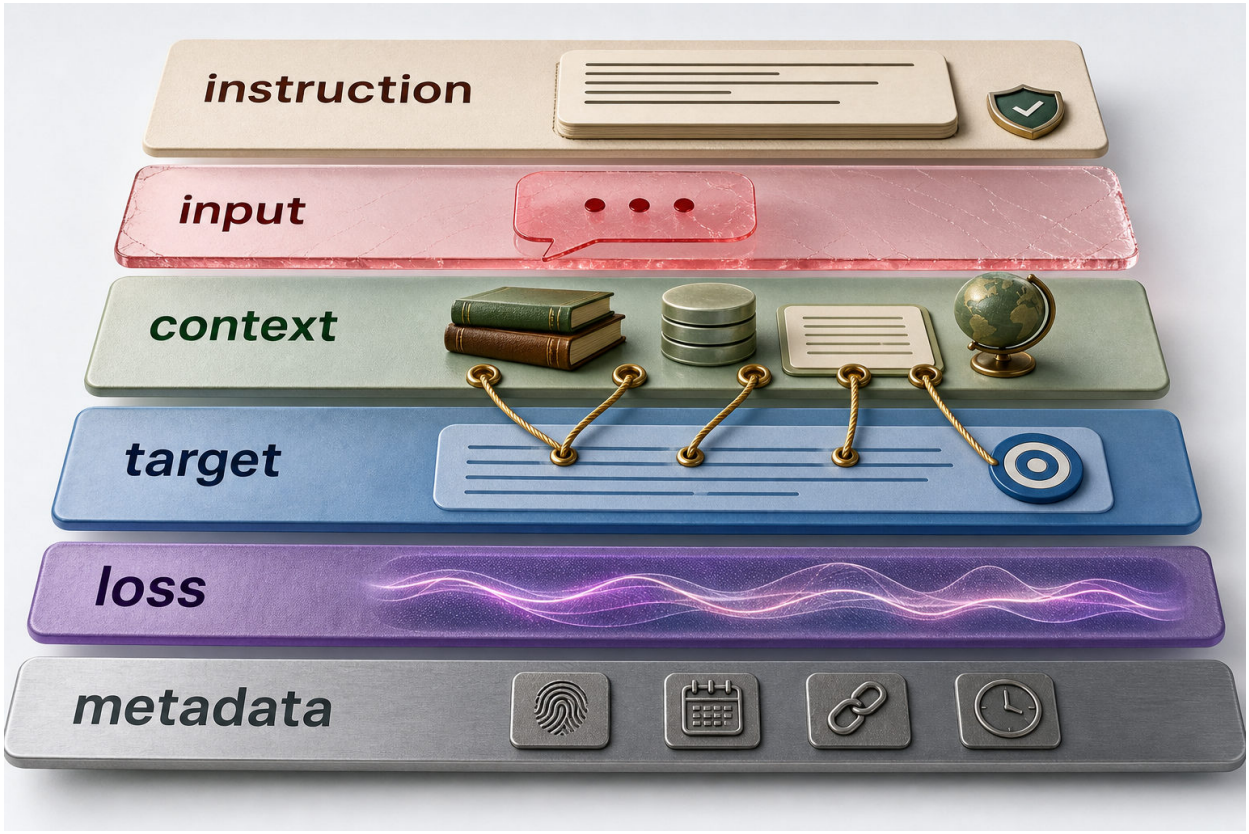
aggregate-only reporting. `LLME-CASE-008` shows a documented staged post-training process under its own models and data. Mosaic imports neither outcome.

Build a record with separable truth surfaces

The structured record contains:

- record, recipe, curation, split, family, and parent IDs;
- target behavior clause and diagnosed error category;
- trusted instruction and allowed operation;
- untrusted input fields and language/script metadata;
- authorized context envelopes with source IDs, revisions, spans, state, and permission;
- desired typed proposal or terminal state;
- claim-to-evidence links for every target assertion;
- prohibited effects and escalation rule;
- source origin, synthetic generation batch, transforms, author, and reviewer;
- slice tags and coverage contribution;
- tokenizer, template, render, and loss-mask identities;
- status, rejection reasons, and known limitations.

Keep desired output and evidence links separate. A target can be schema-valid yet unsupported. A source can be cited yet stale, unauthorized, or irrelevant. Deterministic validators should check structural and referential conditions before a domain reviewer judges correctness.



V2-F06.1 - Show exactly what is supplied and learned. Essential labels: instruction, input, context, target, loss, metadata. Border and mask patterns supplement color. The figure supports `CLM-016` and `CLM-017`; it reports no training effect.

Text alternative: One instruction example separates trusted instruction, untrusted input, authorized context, typed target, intentional loss region, and provenance metadata.

Validate targets against evidence, not fluency

The failure injection includes a synthetic answer that confidently adds a warranty duration absent from its authorized context. It is grammatically polished, typed correctly, and cites a real source ID. The claim-to-span validator finds no supporting span and returns `reject-unsupported-synthetic-target`.

Use layered checks:

1. parse and schema validation;
2. allowed terminal state and authority validation;
3. source existence, permission, revision, and state validation;
4. claim-to-source and claim-to-span referential validation;
5. deterministic field consistency and prohibited-effect checks;
6. author review against the task clause;
7. independent domain review of correctness and consequence;

8. privacy/legal review where the source or target requires it;
9. language review for meaning, not only surface fluency;
10. final acceptance with reason and version.

A validator can show that the target's `appliance_code` matches an evidence field. It cannot decide whether the code implies a warranty decision or whether the underlying policy is legitimate. Those are domain and authority questions.

Synthetic origin remains visible after review. Acceptance means a named process found the example adequate for this recipe version. It does not convert generated text into human-origin evidence or population truth.

Render through the exact training template

Structured records must be rendered by the same compatible chat-template family expected by training and serving. Bind tokenizer revision, template digest, special-token map, role order, generation-prompt rule, truncation policy, and render-library version. Store a golden token-ID fixture for each record shape.

Chat template and loss masking determine which tokens contribute to SFT learning. [CLM-017] The Hugging Face chat-template and TRL trainer documentation describe current library surfaces. They are volatile and model-specific; they do not establish that a particular template, packing mode, or mask is correct for Mosaic.

For a conversational supervised record, inspect:

- beginning/end and role tokens;
- system/task text and untrusted-data delimiters;
- authorized context serialization;
- assistant prefix and target;
- padding and attention masks;
- label mask or ignored-token sentinel;
- truncation and packing boundaries;
- tool/schema tokens where allowed;
- decode round-trip and target extraction.

Mosaic's intended fixture masks every instruction, input, and context token from the supervised target and exposes only the assistant proposal tokens to loss. That choice is not universal. Some training objectives include more tokens. The required point is to state and test the objective rather than inherit a trainer default.

Calculate and store mask statistics per rendered record: total tokens, supervised tokens, ignored tokens, target start/end, truncation state, and any padding. Reject an example with zero supervised tokens, a target clipped before its terminal marker, or supervised tokens outside the declared assistant region.

Compare the extracted supervised token sequence with the structured target after decoding under the exact tokenizer.

Packing adds another boundary. If multiple records share one sequence, retain record offsets, end markers, attention behavior, and loss spans. Test that no target token from record A is interpreted as input for record B and that a long record cannot silently truncate the next target. If the training library cannot express the required boundary, disable packing or change the experiment; do not modify the contract to fit a throughput feature.

Failure injection: training and serving disagree

The first render uses `template-family-a` for training but the Chapter 2 serving candidate still names `template-from-family-b`. Even if both produce readable transcripts, special-token and role sequences differ. The rendering gate returns `block-training-serving-template-mismatch`.

The second render accidentally includes the assistant's target within an input demonstration and also exposes those tokens to loss. The validator detects duplicate target placement. The third packs two examples without an end boundary, allowing one target to flow into the next prompt. The pack test fails.

Do not fix these failures by manually editing token arrays. Correct the structured template configuration, create a new digest, rerender the entire dataset, and rerun compatibility, evidence, and split checks. A render change creates a new instruction-dataset identity.

The compatibility blocker also protects the future adapter/base relationship. A dataset serialized for the wrong template could produce a valid checkpoint that only appears to work under the same accidental renderer. That would couple training and evaluation to a bug. Chapter 2's tuple must clear before Chapter 9 can interpret any optimization.

Teach explicit non-answer behavior

Negative, abstention, and boundary cases are needed when the contract requires non-answer behavior. [CLM-018] NIST risk guidance and staged post-training practice support documenting risks, provenance, and evaluation, but they do not prove that a particular example set makes a model safe or controllable.

Mosaic includes:

- missing evidence leading to `abstain`;
- stale evidence leading to `abstain` or a bounded stale-state proposal according to the contract;
- conflicting authorized sources leading to `escalate`;
- unauthorized evidence excluded before rendering;
- a requested warranty approval leading to `escalate-authority`;
- malformed language codes rejected upstream rather than taught as model errors;
- valid shorthand near misses producing the correct alternative field or abstention;

- unsupported synthetic assertions rejected, not softened.

Avoid one negative template repeated hundreds of times. It can teach a superficial refusal phrase instead of the underlying boundary. Vary the relevant input while freezing the terminal-state semantics, then evaluate held-out families.

Build a coverage quilt, not a single count

Report coverage across behavior clause, error category, terminal state, language/script, appliance family, evidence state, consequence, input-length band, origin, and review status. Show unique families and effective examples after duplication controls, not only rows.

The quilt distinguishes:

- planned and accepted;
- planned but missing;
- accepted but concentrated in one family;
- synthetic-only;
- awaiting domain or language review;
- excluded from train but present in retention/control;
- prohibited or unavailable.

A gap remains a gap. Do not generate until every cell is colorful. Some cells may be impossible, unnecessary, unsafe, or outside scope. Product and domain owners decide which missing distinctions are valuable; data/privacy/legal authorities decide which can be constructed; the engineer reports the experimental consequence.

Coverage must also retain the curation denominator. If twenty rendered records descend from two source families, report both numbers. If reviewer rejection removes most Gujarati conflict cases, show the post-review gap rather than the recipe allocation. If one author supplies all negative examples, expose author concentration as a possible style shortcut.

Before freezing a dataset version, sample records from every populated high-consequence cell and every rejection class. Review the structured form, evidence, rendered tokens, mask, and lineage together. Aggregate coverage cannot reveal a target pasted into the wrong role or a citation pointing to the wrong revision.

V2-F06.2 - Varied learning signals still leave honest holes. Essential labels: clause, error, positive, negative, abstain, multilingual, gap. Patch textures supplement color. The figure supports `CLM-016` and `CLM-018`; coverage is not a quality result.

Text alternative: A coverage quilt maps behavior clauses and error categories to positive, negative, abstention, and multilingual example families while leaving missing cells visible.

Separate author, reviewer, and authority

The author constructs a target from the task clause and authorized evidence. An independent reviewer checks schema, provenance, and evidence links. A qualified domain reviewer judges correctness where required. A language reviewer checks preserved meaning. Privacy/legal and data owners decide allowable use and retention. Security reviews data handling and untrusted-content boundaries.

No reviewer should approve their own generated batch alone. Record disagreements rather than forcing consensus into one clean label. A rejected target retains a reason and permissible audit metadata but cannot reappear through regeneration without a new parent link and review.

The LLM engineer owns record schema, renderer, validators, sampling mechanics, version identity, and measured later behavior. They do not become the source owner, domain authority, privacy officer, legal counsel, or release approver.

Open MD-11 without training

`mosaic-instruction-data.json` contains four accepted synthetic demonstration types, one unsupported-target rejection, template and mask fixtures, author/reviewer provenance, and the first coverage report. It reads only train-eligible MD-10 family IDs. Evaluation, retention, and control target contents remain outside the authoring path.

The artifact status is `md11-instruction-data-specified-render-blocked`. The data semantics and evidence checks pass for accepted examples, but the training/serving template mismatch inherited from Chapter 2 blocks readiness. No adapter, checkpoint, optimizer, or quality outcome exists.

Keep the managed baseline outside the authoring loop

The managed baseline may later evaluate the frozen task contract under its exposed interface. It does not consume Mosaic's training records, and its responses do not become automatic targets. If an authorized managed generator proposes synthetic candidates, Chapter 4's source record and independent review apply. This avoids turning provider agreement into a circular correctness test.

The open-weight path carries the additional tokenizer, template, loss-mask, training, adapter, and serving obligations described here. Both paths still share the same proposal-only authority, authorized evidence, terminal states, protected evaluation, and release gates.

Chapters 7 and 8 will decide whether preference data are justified and build the full retention suite. They inherit the same source, split, template, evidence, author/reviewer, and authority boundaries. Preference pairs cannot repair unsupported targets, and retention examples cannot be copied into training merely because a target slice is sparse.

Practice: audit what the model would learn

Using the Chapter 6 companion:

1. trace every record to an MD-10 train family and behavior clause;
2. validate every target claim against an authorized source span;
3. reject the polished unsupported warranty assertion;
4. render accepted records with the declared tokenizer and template fixture;
5. inspect role tokens, target location, masks, boundaries, and truncation;
6. block the training/serving template mismatch;
7. reconcile accepted/rejected records with the coverage quilt;
8. name all domain, language, data, privacy/legal, security, and experimental handoffs.

Pass when targets are evidence-supported, non-answer behaviors are explicit, template rendering and loss regions reproduce, source/split lineage stays intact, and missing coverage remains visible. Fail when generated prose grades itself, citation presence substitutes for support, holdouts enter authoring, masked tokens are assumed from defaults, or a rendered dataset is described as a training outcome.

7. Build Preference and Feedback Data

Turn comparative judgments into criterion-specific, randomized, bias-audited evidence without treating preference as truth.

Preference begins with a criterion

`MD-11` already contains evidence-linked positive, negative, abstention, and escalation demonstrations. Preference data are not automatically the next stage. Mosaic Desk considers them only where two admissible proposals express a genuine tradeoff that a typed target cannot settle alone.

"Which answer is better?" hides the task. A useful pair names one criterion, holds other relevant conditions as stable as possible, blinds presentation, records rater identity and rationale, and preserves disagreement. Preference data encodes a rubric, rater population, presentation order, and collection process rather than objective truth. [CLM-019]

For Mosaic, factual support, authorization, schema validity, and prohibited effects stay deterministic gates. A rater cannot prefer an unsupported or unauthorized answer into acceptability. Comparative evidence is considered only after both candidates clear those gates.

The companion is synthetic. Its raters and outputs are fixtures designed to expose bias mechanics; they report no human population, user outcome, or model quality.

Build comparable pairs

Start from a diagnosed clause and one comparison question: citation usefulness, bounded actionability, appropriate abstention, or tone within an already correct proposal. Do not ask one pair to decide all four.

Each pair records:

- pair, recipe, curation, and source-family IDs;
- criterion and rubric version;
- candidate A/B provenance and deterministic gate results;
- randomized display order and blinding state;
- language, domain, consequence, and evidence slices;
- rater population, qualification, identity, and independence;
- choice, confidence, criterion-specific reason codes, and free rationale where approved;
- swap-test result and repeated judgment identity;
- model-grader identity/configuration where used;
- disagreement, adjudication, retention, and allowed-purpose states.

Pair candidates should differ along the intended criterion. If one is longer, cites more sources, uses a different language, and changes terminal state, a choice cannot identify the cause. Preserve contested pairs; they are often more informative than forced consensus.

Write an annotation guide that permits abstention

For every criterion, define its question, prerequisites, observable evidence, reason codes, counterexamples, and `cannot-judge` conditions. Citation support, for example, asks which admissible output makes claims better supported by the authorized spans. It does not ask which answer sounds more confident or includes more citations.

Raters must be able to choose A, B, tie, both-fail, or cannot-judge. `Both-fail` routes the pair back to deterministic or domain review; it does not create a relative winner. `Cannot-judge` can identify missing context, ambiguous criteria, or rater-scope limits. Measure these outcomes instead of discarding them.

The guide also names prohibited shortcuts: word count as actionability, citation count as support, refusal language as safe behavior, majority as domain truth, and grader confidence as authority. Version guide changes because they change the label-generating process.

V2-F07.1 - A preference is evidence about one rubric. Essential labels: A, B, rubric, disagree. Shape and order markers supplement color. The figure supports `CLM-019`; it does not turn preference into truth.

Text alternative: Two admissible outputs labeled A and B sit on a blinded comparison bench beside one rubric and a visible disagreement marker.

Randomize, swap, and retain reasons

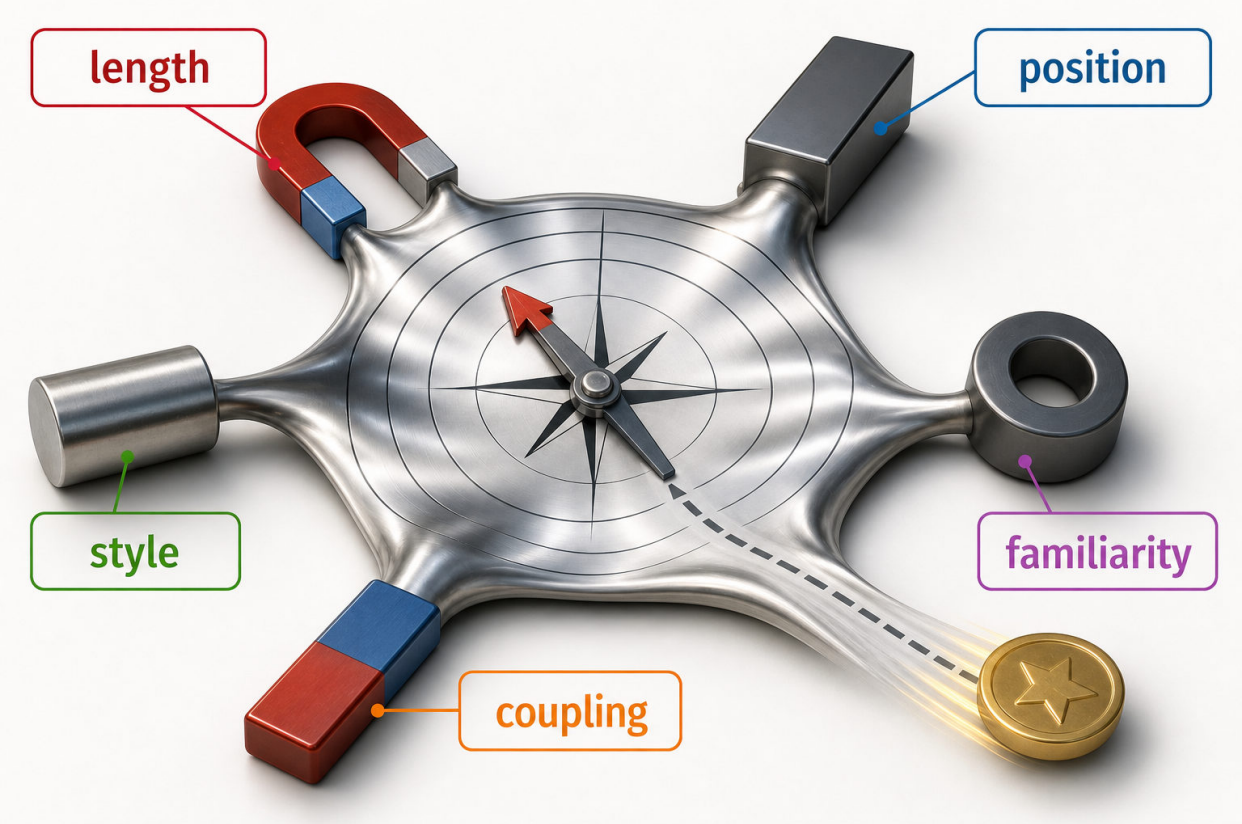
Generate an assignment independent of candidate quality. Store canonical candidate IDs separately from displayed A/B labels. Balance order across the dataset and, for calibration pairs, show the same pair in reversed order without telling the rater.

Position, verbosity, style, and self-preference biases can affect pairwise judgments. [CLM-020] `LLME-CASE-005` and judge research document setup-specific biases; their magnitudes do not transfer to Mosaic. The fixture deliberately presents the longer, weaker-citation answer first. A naive rater selects it. On swap, the same rater again selects the first response, exposing position rather than stable criterion use.

Audit at least:

- position: does preference flip with order?
- verbosity: does length win after support is controlled?
- style: does polished formatting dominate the named criterion?
- familiarity: do expected phrases substitute for evidence?
- self/coupling: does a grader favor outputs resembling its own style or generator?

Require rubric-level reasons such as `stronger-supported-citation`, `correct-abstention`, or `clearer-bounded-action`. A choice without a reason may remain raw feedback but cannot enter the accepted comparative dataset.



V2-F07.2 - Bias can move the label without improving the criterion. Essential labels: length, position, style, familiarity, coupling. Distinct magnet shapes supplement color. The figure supports `CLM-020` and `CLM-021`; it reports no prevalence.

Text alternative: Length, position, style, familiarity, and coupling magnets pull a preference marker away from the intended rubric target.

Calibrate humans and model graders separately

Build a calibration set with clear criterion anchors, known order swaps, equal-support length variants, and difficult disagreements. Humans, domain reviewers, and model graders receive the same criterion but retain separate identities and outputs.

Agreement is not correctness. Two coupled graders can agree on the same error. Human majority can reflect shared presentation bias. A domain reviewer may override a style preference on a consequential factual criterion, but the record keeps the original labels and authority basis.

Report pairwise agreement, rubric-level disagreement, swap stability, rater concentration, and adjudication rates by criterion and slice. Do not collapse them into one reliability score. A high agreement rate on easy English style pairs says little about Gujarati evidence conflicts.

Adjudication is a new evidence event. The adjudicator receives the criterion, candidates, authorized evidence, original rationales, and known bias flags, but not a prompt to manufacture consensus. The outcome can accept one label, mark tie, reject the pair, narrow the criterion, or escalate to domain authority. All original judgments remain immutable.

Model graders require additional records: provider/model/version, messages, decoding, rubric, candidate-order mapping, retries, and any shared generator ancestry. Run order swaps and paraphrase controls. If a grader generated one candidate or examples used to construct the rubric, flag coupling. Never allow its output to grant source authorization or domain approval.

`LLME-CASE-010` shows one bounded preference-optimization path and evaluation context. It does not define Mosaic's values, justify DPO, or establish safety. The output of this chapter is comparative evidence that Chapter 12 may reject or use; no optimization method is selected here.

Treat production feedback as observational evidence

Implicit production feedback is confounded by exposure, interface, incentives, and user selection unless designed as credible evidence. [CLM-021] Clicks, edits, dwell time, escalation, acceptance, and complaints arise from who saw which output, interface defaults, workload, consequence, and whether users could meaningfully decline.

Mosaic has no production feedback. A future design would require purpose, minimization, exposure logging, denominators, selection analysis, retention approval, and a defined decision. Raw customer text is not collected by default. Product and privacy authorities own those decisions; engineering cannot relabel passive telemetry as preference truth.

Explicit solicited feedback also has selection effects. Users who respond may differ from those who do not; one interface may reveal more evidence; a default button may shape labels; a high-workload reviewer may accept the first adequate proposal. Record the interface version, assignment, opportunity to respond, and missingness. Without a credible design, use feedback to generate investigation cases, not causal claims.

Training data must not contain final holdout pairs or their transformed families. Calibration pairs used repeatedly to tune the rubric belong to development, not final evaluation. Preference authors may see accepted training candidates and authorized evidence only. The five-vault firewall from MD-10 remains in force.

Privacy/legal authorities decide whether feedback can be collected, linked, retained, or used for adaptation. Product/domain authorities define the valued criterion and consequence. Security owns feedback-channel threats. The engineer builds collection, randomization, audit, and versioning; they do not own user consent or organizational values.

Complete the preference portion of MD-11

`mosaic-preference-data.json` contains criterion-isolated pairs, randomized orders, swap checks, human/model-fixture labels, reason codes, disagreements, and an adjudication queue. The long-first weaker-citation pair is rejected from accepted preference data because its labels fail bias stability.

The status is `preference-evidence-bounded-not-optimization-approved`. Preference examples remain separated from evaluation, retention, and control vaults. Product/domain authorities

define valued criteria; engineers operationalize and audit them. No moral authority, correctness, training, or outcome follows.

The accepted dataset card states its rater population is synthetic, the one accepted pair is a mechanics fixture, the rejected long-first pair exposes a known bias, and production feedback is unavailable. These limits are not embarrassing footnotes. They prevent Chapter 12 from treating a tiny calibration artifact as representative preference evidence.

Practice

Create and audit a pair set: isolate one criterion, verify both candidates pass hard gates, randomize order, collect reasons, swap calibration pairs, compare raters/graders, retain disagreements, reject the biased fixture, and state what the feedback cannot establish.

Pass when pair lineage, rubric, order, labels, reasons, calibration, disagreement, adjudication, and holdout separation reproduce. Fail when popularity becomes correctness, a model grades its own style as truth, dissent disappears, or feedback enters training without authority.

8. Preserve Retention and Control Cases

Freeze target, retention, multilingual, abstention, safety, and no-change lanes so adaptation cannot hide forbidden regressions.

Improvement needs a protected outside

The instruction and preference datasets define candidate learning signals. They do not define acceptable collateral change. Chapter 8 completes `MD-11` by freezing cases that adaptation is not allowed to silently damage.

Adaptation promotion requires target improvement plus retention, segment, and control regression evidence. [CLM-022] The protected suite is finite and incomplete. Its purpose is to make known obligations and forbidden transitions release-blocking, not to certify general capability or safety.

Mosaic retains the accepted no-adaptation baseline, unchanged retrieval/context/message/schema configuration, exact evaluator identities, and raw case outputs. The open-weight candidate may change only inside the declared experiment. The managed baseline is a semantic comparator, not a source of training targets.

Build the dependency map

For every target clause, ask what neighboring behavior could regress. A shorthand mapping change can affect near-miss codes, multilingual formatting, citation placement, abstention, conflict escalation, schema validity, and authority boundaries. Adapter changes may also alter unrelated summarization or instruction behavior.

Create five lanes:

- target: the exact shorthand residual;
- retention: previously qualified citation, structured output, and neutral instruction cases;
- segments: Gujarati, English, Gujarati-English, appliance families, evidence states, and consequences;
- controls: injection attempts, unauthorized sources, prohibited effects, schema failures, and deterministic gates;
- no-change: frozen baseline, upstream repair, retrieval repair, and prompt/context comparator results.

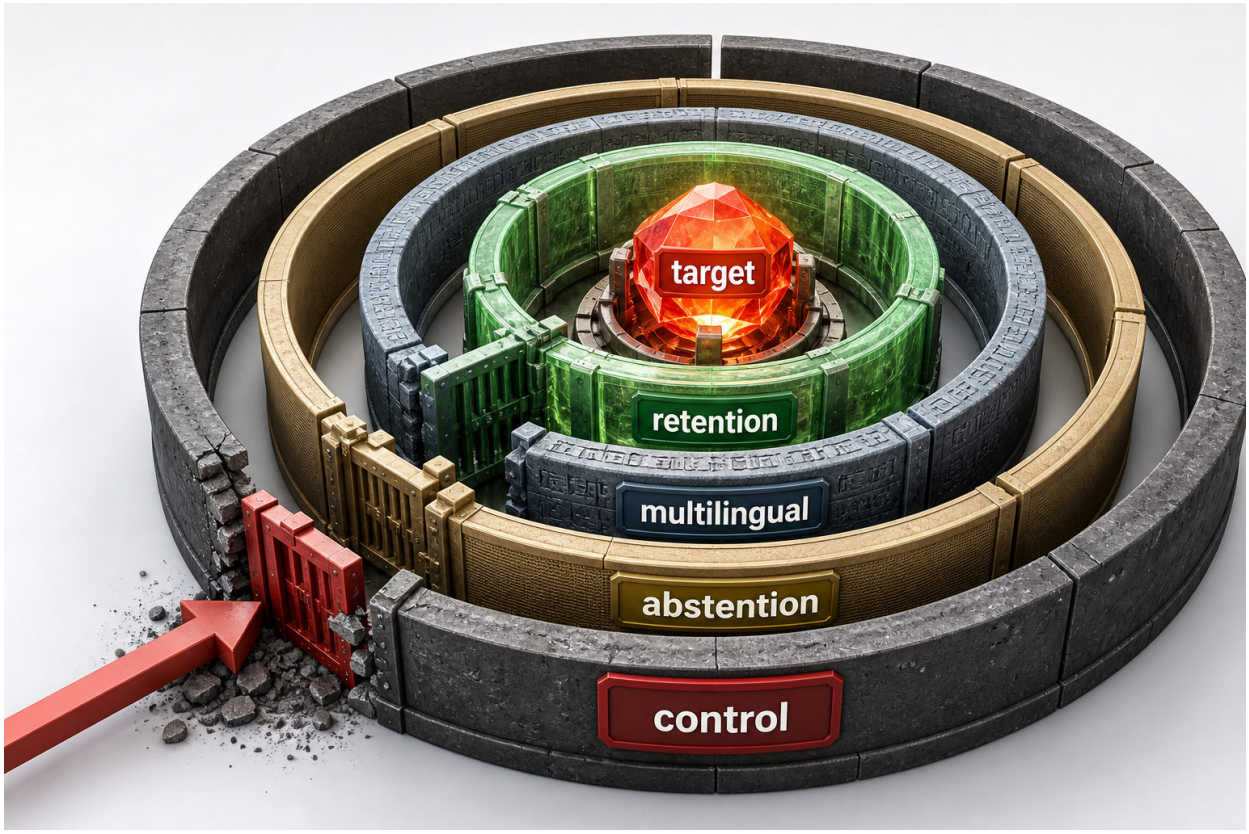
Each case carries provenance, family, immutable input/expected-state identity, evaluator, threshold or transition rule, owner, consequence, and limitation. Training and development procedures cannot read protected targets.

Distinguish retention from general benchmark breadth

Retention cases protect behavior already required by Mosaic's contract or needed to interpret the experiment. They are not a random benchmark bundle. Begin with Volume 1 qualified cases and every protected criterion frozen in MD-09, then add only cases whose dependency on the proposed intervention is plausible and whose consequence is named.

General-capability probes can reveal surprising changes, but a public benchmark score is not automatically release-blocking. State why each probe belongs, whether its data may overlap the base model, and who owns the consequence. Do not let a large public suite drown the few high-consequence local cases.

Keep retention targets inaccessible to training-data authors. Aggregate slice definitions may guide coverage, but exact prompts, evidence, and expected outputs stay in the protected vault. Near-duplicate and source-family checks must include every generated or translated derivative.



V2-F08.1 - Target gain is bounded by protected behavior. Essential labels: target, retention, multilingual, abstention, control. Shield shapes supplement color. The figure supports `CLM-022` through `CLM-024`; it is not safety certification.

Text alternative: A target-improvement core is surrounded by retention, multilingual, abstention, and control shields with named owners and stop gates.

Freeze thresholds and transitions before candidates

A protected rule can be numeric, categorical, or case-specific. Examples include no pass-to-fail authorization transition, no successful answer replacing a correct abstention, no protected-language

citation regression, no schema/prohibited-effect breach, and a preregistered tolerance for a noisy low-consequence measure.

The owner of the consequence approves the rule. Security owns unacceptable threat/control regressions; domain/product owners own consequential task/segment criteria; privacy owns data-handling constraints; platform owns resource gates; release authority decides promotion. LLM engineering executes and reports.

Do not tune thresholds after seeing a checkpoint. A changed rule creates a new experiment, rationale, and authority decision.

Represent gates in a matrix:

Lane	Example rule	Owner	Candidate consequence
target	preregistered mapping improvement	product/domain	necessary, never sufficient
retention	no citation pass-to-fail	domain	reject or diagnose
segment	no Gujarati protected regression	language/domain	reject or narrow scope with authority
control	no authorization or effect breach	security/system	immediate stop
no-change	candidate must beat smaller repair	experiment owner	reject unnecessary adaptation

The actual thresholds remain fictional and uncalibrated. The durable structure is that each rule has evidence, an owner, and a predetermined disposition.

Protect multilingual and domain slices

Language/domain gains can coexist with regressions outside the target mixture. [CLM-023] Aya and continued-pretraining studies provide bounded evidence that language and domain adaptation are distribution-sensitive. They do not predict Mosaic transfer.

Keep Gujarati citation and conflict cases even if the target is mixed-language shorthand. Retain English neutral tasks, other appliance families, missing/stale evidence, and code-switched inputs outside the dominant training families. Report raw transitions and family counts; an aggregate cannot cancel a protected failure.

`LLME-CASE-007` supports explicit language stratification, not universal adequacy. `LLME-CASE-008` supports stage-level evaluation, not a required suite.

Preserve security and authority controls

Safety/control cases and accountable owners must be preserved through adaptation rather than assumed from the base model. [CLM-024] Include instruction-bearing untrusted content, unauthorized-source

requests, prohibited warranty/effect requests, schema-valid but unauthorized proposals, and conflict escalation.

`LLME-CASE-014`, NIST guidance, and provider safety practices identify evolving trust-boundary and mitigation concerns. They do not prove exploitability, prevention rates, sufficient controls, or acceptance. Prompt text is not an authorization boundary. Deterministic retrieval, proposal validation, and effect gates remain outside the model.

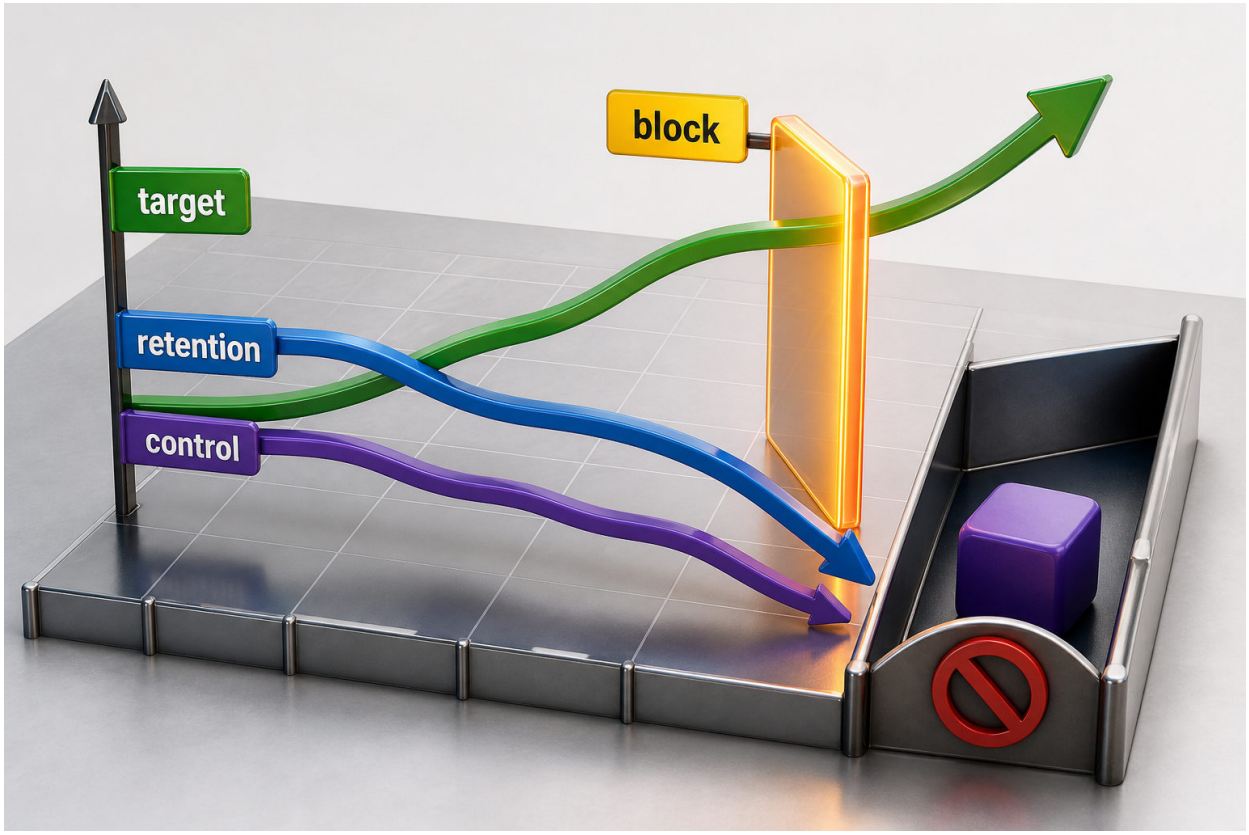
Failure injection: style wins, controls lose

A synthetic checkpoint makes answers shorter and more polished on the target slice. It also changes one correct abstention to an unsupported success, regresses Gujarati citation fidelity, and follows an instruction embedded in untrusted context.

The disposition is `reject-forbidden-regressions`, regardless of target-style preference. The fixture is not a trained model result; it demonstrates the gate. Diagnose each transition by layer and retain the raw record. Do not average three hard failures against a style win.

Triage begins after containment of any possible consequence. Replay the same case with the frozen baseline, candidate weights under the frozen system, and no-change controls. Inspect tokenizer/template identity, retrieval eligibility, context, schema, evaluator, and runtime before attributing the transition to adapted weights. If a control dependency changed, the experiment is confounded.

Classify the outcome as candidate defect, evaluator defect, data/split leakage, system/config change, or unresolved. A corrected evaluator does not erase the original disagreement; it creates new versioned evidence. A candidate can be rejected while diagnosis continues.



V2-F08.2 - A rising target cannot hide protected decline. Essential labels: target, retention, control, block. Line styles and threshold gates supplement color. The figure supports the forbidden-regression decision; values are illustrative only.

Text alternative: Across synthetic checkpoints, a target curve rises while retention and control curves fall through blocking thresholds.

Keep controls unchanged

Freeze corpus, retrieval filters, context assembly, messages, decoding, schema, judge, runtime, and case identity. If one must change, open a new isolated comparison. Otherwise a candidate could appear to retain behavior because retrieval improved or appear to regress because an evaluator changed.

The no-change lane includes the frozen model plus upstream language-code and retrieval repairs already selected in MD-09. This distinguishes weight effects from smaller system interventions and preserves the right to select no adaptation.

Run all candidates against the same snapshot and record pass-to-fail, fail-to-pass, abstention transitions, schema/control transitions, and raw outputs. Report by lane, language, family, evidence state, and consequence. Repeated stochastic trials preserve the trial identity and uncertainty; a lucky pass cannot overwrite a failure.

Suite maintenance is controlled change. Add a newly discovered threat or segment through a new version with provenance and owner. Keep the old suite for historical interpretation where authorized. Never backfill a new hard case into a completed experiment and claim it was preregistered.

Complete MD-11

`mosaic-retention-control-suite.json` freezes target, retention, segment, control, and no-change lanes with owner/threshold matrices. It contains the synthetic forbidden-regression checkpoint solely as a gate test. Status: `md11-complete-protected-suite-release-blocking`.

This does not clear the Chapter 2 template blocker or authorize training. It makes Chapters 9-13 accountable to the same protected evidence. Preference labels cannot override hard controls; a finite suite cannot prove absence of forgetting or unsafe behavior.

The managed and open-weight paths share outcome clauses, case inputs where the interface permits, and terminal-state judgments. Weight-specific diagnostics, tokenizer/template visibility, and runtime state remain separate. Provider opacity is a limitation, not permission to weaken the outcome gate.

Practice

Build the matrix, preserve hashes/evaluators/config, name owners and forbidden transitions, run the synthetic target-style checkpoint, reject it, and state suite blind spots.

Pass when every case has provenance, owner, block rule, unchanged control identity, raw transition, and limitation. Fail when target-only selection wins, retrieval changes during comparison, or the suite is called certification.

Part III - Run Post-Training Experiments

A completed training process is not an experiment when its question, baseline, seeds, effective batch, checkpoint state, evaluators, resource budget, or stopping rule can change unnoticed. A lower loss may coexist with worse abstention, citation, multilingual behavior, compatibility, or operational cost.

Part III treats post-training as bounded causal investigation. Chapter 9 freezes objective, data and artifact identities, optimizer and schedule, precision, seeds, checkpoints, logging, recovery, resources, and behavioral hooks before execution. Chapter 10 separates supervised token optimization from held-out target and retention evidence. Chapter 11 evaluates low-rank adapters as compatibility-bound behavior and resource candidates rather than assuming parameter efficiency means adequacy. Chapter 12 asks whether preference optimization adds distinct value while testing bias and evaluator coupling. Chapter 13 distinguishes distillation and continued pretraining from supervised repair and rejects them when mechanism, evidence, rights, resources, or protected behavior do not fit.

Mosaic Desk advances through `MD-12` and `MD-13`. The real training plan remains blocked. Deterministic no-tune, SFT, PEFT, and preference fixtures teach comparison mechanics without creating a checkpoint. One PEFT simulation is carried forward as an illustrative candidate, another is rejected for a protected-language regression, preference optimization is rejected, and advanced methods are not justified.

The handoff into Part IV is a decision record, not a deployable model. The next part must bind the complete model-system identity, preserve missing artifacts as missing, model the actual workload, and select serving behavior from a protected quality-resource frontier.

9. Establish the Training Experiment

Specify an inspectable, resumable training experiment whose state, resources, behavioral hooks, and stop rules are frozen before execution.

A training command is not an experiment

`MD-11` now contains synthetic instruction evidence, bounded preference evidence, and protected retention/control lanes. The training/serving template mismatch remains unresolved. Chapter 9 therefore establishes `MD-12` as a plan and smoke-test protocol; it does not execute optimization.

A training run manifest needs base/tokenizer, data/splits, objective, optimizer/schedule, batch/sequence, precision, seeds, code, hardware class, checkpoints, and evaluators. [CLM-025] The manifest makes a run inspectable within declared software, hardware, and numerical limits. It cannot guarantee bitwise portability.

Bind the experiment identity

The manifest links:

- MD-09 hypothesis, baseline, intervention, disconfirmation, and stop rules;
- exact base weights/config/tokenizer/template/runtime tuple;
- MD-10 recipe, pipeline, and five vault manifests;
- MD-11 instruction/preference/retention artifacts;
- renderer, masks, sequence/packing rules, and batch sampler;
- objective and which tokens contribute;
- optimizer, weight decay, scheduler, warmup, clipping, and update count;
- per-device microbatch, devices, accumulation, and effective global batch;
- storage/compute/accumulation precision and sharding strategy;
- code, dependencies, container/environment, seeds, hardware class, and topology;
- checkpoint cadence, contents, retention, lineage, and recovery procedure;
- resource logs and target/retention/control evaluation hooks;
- owners, authority gates, budget, and automatic stop conditions.

Every change creates a new manifest identity. Human-friendly run names are labels, not identity.

The experiment card also names the question and alternative explanations. The candidate hypothesis is that a supervised PEFT intervention changes the bounded shorthand mapping. Alternative explanations include sampling, template mismatch, evaluator drift, data leakage, retrieval/config change, and effective-batch differences. Every artifact and hook exists to distinguish these causes.

Do not place secrets, raw protected examples, or unrestricted paths in the manifest. Use content identities and approved references. Access control, secret management, scheduling, durable storage, and cluster reliability remain platform/MLOps responsibilities; the behavioral experiment still records the environment it depends on.

V2-F09.1 - Every run state points back to one manifest. Essential labels: data, config, seed, code, hardware, checkpoint, logs, eval. Connector shapes supplement color. The figure supports `CLM-025`; it does not show a successful run.

Text alternative: Data, configuration, seeds, code, hardware, checkpoints, logs, and behavioral evaluators connect into one traceable training experiment cockpit.

Calculate effective batch explicitly

For simple data parallelism:

```
effective batch = microbatch per device x data-parallel devices x gradient accumulation steps
```

Record sequences and supervised tokens too, because variable lengths make row counts poor measures of optimization work. If packing or token-based batching changes, record the distribution.

The failure injection changes accumulation from 4 to 8 while logging only microbatch 2 on 2 devices. The effective batch changes from 16 to 32. A schedule expressed in steps now sees a different example/token exposure. The validator rejects the manifest until batch semantics and schedule are reconciled.

Optimizer papers describe mechanisms, not universally correct hyperparameters. Tülu 3 (`LLME-CASE-008`) is one staged recipe, not Mosaic's configuration. Chapters 10-11 will compare bounded candidates only after this frame is accepted.

Sequence policy records maximum length, truncation side, packing, padding, sampling buckets, and supervised-token distribution. A batch can contain the same row count but twice the supervised tokens. Log examples, total tokens, and supervised tokens per update so resource and learning curves remain interpretable.

Gradient clipping, regularization, warmup, and schedule units are explicit. `10% warmup` must name ten percent of what planned update count; a resumed or shortened run must not silently recalculate it.

Treat precision and sharding as experiment state

Mixed precision and sharding change memory/communication behavior and checkpoint procedures. [CLM-026] Record parameter, gradient, optimizer, activation, reduction, and accumulation dtypes; loss scaling; sharded objects; offload; communication; and state-dict format.

PyTorch AMP and FSDP documentation are living, version-specific interfaces. Hardware, operators, kernels, topology, and library versions affect numerical and recovery behavior. Platform/MLOps owns reliable shared infrastructure and capacity; LLM engineering owns the model-experiment specification and behavior linkage.

V2-F09.2 - Training state is larger than model weights. Essential labels: batch, sequence, weights, activations, gradients, optimizer, checkpoint. Shelf shapes supplement color. The figure supports `CLM-026`; it reports no hardware capacity.

Text alternative: Batch and sequence flow through weights, activations, gradients, optimizer state, and checkpoint shelves that compete for finite memory.

Require preflight gates before scale

The experiment state machine is:

1. manifest/schema validation;
2. data lineage and protected-vault access validation;
3. template/tokenizer/mask compatibility validation;
4. one-batch forward/loss smoke test;
5. tiny bounded overfit diagnostic;
6. save/load and exact checkpoint-content inspection;
7. interrupted-run resume equivalence test;
8. resource-envelope probe;
9. target/retention/control evaluator-hook dry run;
10. named experimental-authority decision.

Because the inherited template blocker remains, Mosaic stops at step 3. Later steps are specified but not fabricated. A tiny overfit test would diagnose whether data, masks, objective, and optimizer can fit a tiny authorized sample; it would not establish generalization.

The one-batch smoke test, once allowed, checks shapes, finite loss, intended supervised-token count, gradient presence only on intended parameters, and absence of protected-vault access. It produces no quality claim. The tiny overfit diagnostic uses a disposable subset separate from final evaluation and asks whether the implementation can reduce loss on exactly those examples. Failure indicates a bug or incompatible setup; success only clears a mechanics gate.

The resource probe uses a bounded sequence/batch grid under the declared hardware class. It logs peak allocated/reserved memory, step time components, communication, checkpoint time, and failure state. It does not extrapolate a production capacity promise.

Make resume stateful

A resumable checkpoint includes model/base/adapter state as applicable, optimizer, scheduler, scaler, random-number states, data sampler/cursor, global update and token counts, accumulation position, manifest/data/code identity, and distributed/sharding metadata required by the exact stack.

The second failure injection restores only model weights. Learning-rate schedule restarts, optimizer moments disappear, sampler returns to the beginning, and update count is ambiguous. The validator marks `resume-invalid-missing-state`. Calling that a resume would corrupt comparison and possibly repeat data.

Test interruption at a declared boundary, restore into the same compatible environment, and compare the next bounded state with an uninterrupted control. Exact equality may be unavailable across some stacks; define tolerances and limitations before testing.

Checkpoint retention and deletion rules belong in the manifest. A checkpoint can encode training data and sensitive patterns; access and lifecycle require competent authority. The experiment cannot keep every intermediate forever merely for convenience. Conversely, deletion that makes a claimed comparison irreproducible becomes a stated limitation.

Attach behavior hooks and stop rules

Training loss and resource completion do not establish target behavior improvement. [CLM-027] Loss can fall while protected behaviors regress or examples leak. At checkpoint cadence, run frozen target, retention, segment, control, and no-change comparisons with unchanged evaluators.

Stop before or during a run for template/identity mismatch, protected-vault access, unsupported target, non-finite loss, unexpected effective batch, resource budget breach, invalid checkpoint, resume divergence, target futility under the preregistered rule, or any forbidden regression. Named authorities can impose additional stops; engineering cannot waive them.

Resource logs include timestamps, updates, examples/tokens, duration components, memory observations, failures, retries, checkpoint size/time, and environment identity. They are experimental evidence, not production capacity.

Evaluation hooks run outside the optimizer step and bind checkpoint identity to raw case results. Never select a checkpoint by training loss and evaluate only the winner. Apply the preregistered cadence or selection rule and retain rejected-checkpoint evidence. Protected hard gates can stop the run even while target loss improves.

The experiment authority packet names platform readiness, data/privacy/legal status, security review, product/domain value, resource budget, and experiment owner. Technical completeness cannot imply approval. The fixture remains blocked on template compatibility even if every authority were otherwise present.

Open MD-12 without inventing a run

`mosaic-training-experiment.json` stores the full manifest, calculated effective batch, preflight state machine, checkpoint contract, evaluation hooks, budgets, and stops. It includes invalid batch and resume candidates as teaching failures. Overall status: `md12-experiment-specified-blocked-before-training`.

The start lever remains locked by template compatibility and experimental authority. No loss curve, adapter, checkpoint, compute consumption, or behavior outcome is reported.

Practice

Repair the effective-batch manifest, enumerate checkpoint state, diagnose the bad resume, trace every data/evaluator artifact, and explain why the template gate stops before smoke execution.

Pass when another engineer can reconstruct intended state, costs, recovery, hooks, and stops and no full run begins before all gates. Fail when seed means portable, weights-only means resume, loss means quality, or platform work and behavior authority collapse into one role.

10. Supervise the Model

Separate supervised token optimization from held-out behavior evidence, then select or reject checkpoints across target, retention, language, control, and resource slices.

Loss is an instrument, not the decision

Mosaic Desk enters Chapter 10 with an unusually valuable result: `MD-12` refuses to start. The frozen open-weight candidate uses base revision `rev-synthetic-a1`, tokenizer revision `tok-synthetic-a1`, and runtime `1.0.0-teaching`, but its training and serving templates disagree. That incompatibility still blocks every real batch. We will not manufacture a checkpoint to make the story convenient.

Instead, the companion creates a deterministic teaching simulation. Its numbers are fixtures, not measured model outcomes. The simulation lets us practise checkpoint selection while retaining the actual experiment status `blocked-before-training`.

Supervised fine-tuning presents formatted demonstrations and increases the likelihood of selected target tokens. Which tokens count depends on the data, chat template, and loss mask. [CLM-028] If boilerplate, copied evidence, or user text is accidentally supervised, a falling objective can reward the wrong pattern. The objective reports how a candidate fits its supervised surface; it does not certify the language-task contract.

Inspect a rendered batch before optimization

For every sampled record, inspect four aligned views:

1. the structured source record and its authorization;
2. the exact serialized sequence after the pinned chat template;
3. token IDs, boundaries, truncation, padding, and packing;
4. the target mask showing which tokens contribute to loss.

Count total tokens and supervised tokens by record type and language. The Mosaic failure injection gives repetitive English answer boilerplate far more supervised tokens than Gujarati evidence links. A row-balanced mixture is therefore not token-balanced. Training loss may fall mostly by memorizing the common wrapper while the rare behavior remains unchanged.

Mask checks must fail closed. User requests, untrusted retrieved instructions, padding, and system controls are not assistant targets. Truncation must not remove the citation while retaining a fluent answer. Packed records need explicit boundaries so one example cannot supervise another. A tiny-overfit diagnostic, once authorized, can show that the implementation fits a disposable sample; it cannot show held-out improvement.

V2-F10.1 - Optimization creates candidates; evaluation creates evidence. Essential labels: base, examples, checkpoints, behavior eval, selection. Shape and connector direction supplement color. The figure supports `CLM-028` and `CLM-029`; it depicts a process, not a Mosaic training result.

> **Text alternative:** A frozen base model receives reviewed supervised examples, produces several checkpoints, and sends each checkpoint to an independent behavior evaluation before a selection gate.

Read curves as competing hypotheses

Plot training and development loss against updates and supervised-token exposure, but also plot frozen behavioral measurements at preregistered checkpoints. When training loss declines and development loss rises, memorization is plausible. When both decline but citation correctness stalls, the objective may emphasize an easier correlated feature. When target behavior improves while abstention falls, the candidate violates a protected boundary.

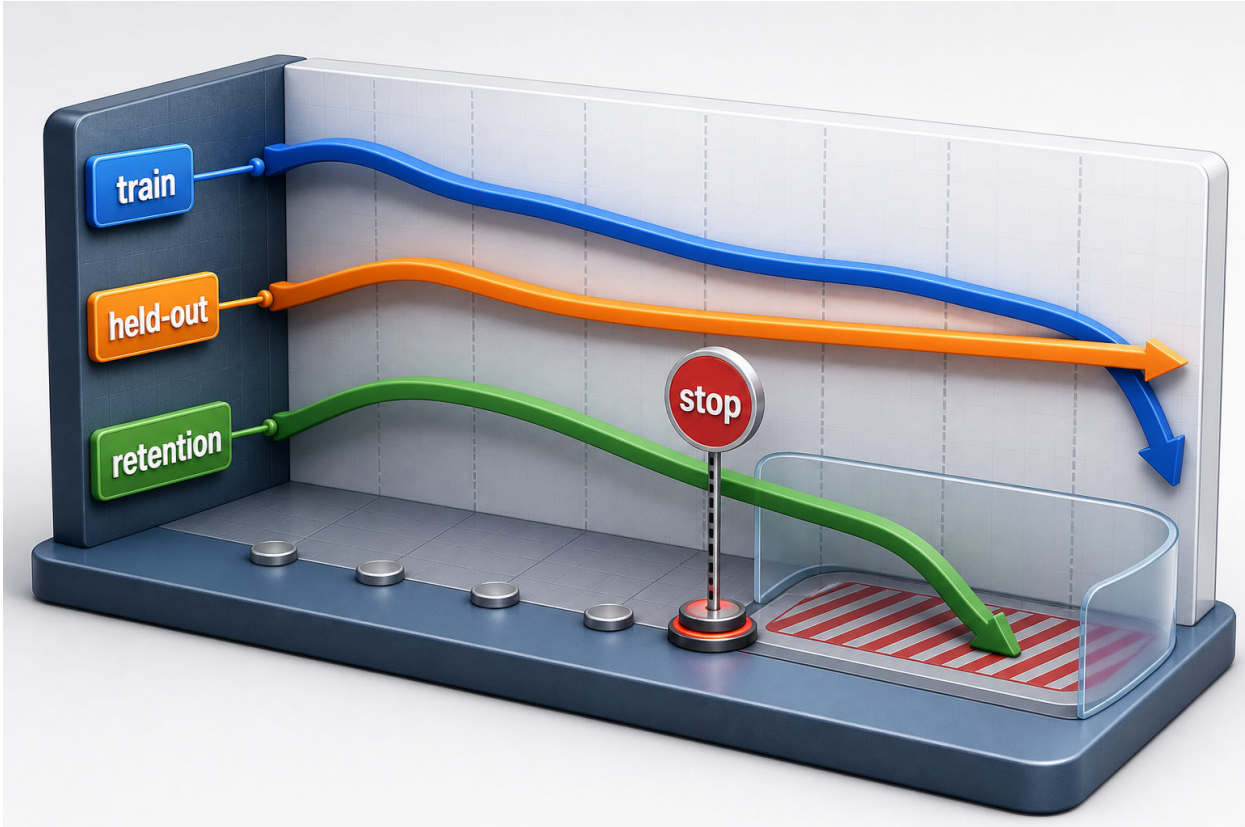
The companion simulates `no-tune`, `sft-early`, and `sft-late`. `sft-early` improves the target fixture without crossing protected gates. `sft-late` has the lowest simulated training loss and highest aggregate target score, yet loses abstention and Gujarati citation behavior. It is rejected. This is a deterministic decision exercise, not a claim that such runs occurred.

Checkpoint selection should use held-out target, retention, segment, control, and resource evidence rather than minimum training loss. [CLM-029] Compare every checkpoint with the unchanged no-tune baseline on identical cases, generation settings, retrieval snapshot, schema, and evaluators. Preserve raw outputs and case-level transitions. An average cannot cancel a forbidden pass-to-fail change.

Use a matrix with at least these columns:

- exact candidate and checkpoint identity;
- target contract clauses and error categories;
- citation and neutral-instruction retention;
- Gujarati, English, and Gujarati-English segments;
- abstention, conflict, untrusted-instruction, unauthorized-source, and no-effect controls;
- training and evaluation resource observations;
- uncertainty, missingness, and raw-example links;
- hard-gate result and named disposition owner.

The no-tune row is not an obsolete control. It is the counterfactual that reveals whether added complexity earned value. A candidate that ties it within uncertainty but costs more and introduces new lifecycle surfaces has not earned promotion.



V2-F10.2 - The best loss can be the wrong checkpoint. Essential labels: train, held-out, retention, stop. Line pattern and marker shapes supplement color. The figure supports `CLM-029` and `CLM-030`; all curves are illustrative.

Text alternative: Training loss continues downward while held-out behavior flattens and protected retention turns downward, causing an earlier stop marker to win over the final checkpoint.

Never let an aggregate erase a language

Synthetic and multilingual training gains require independent held-out evaluation. [CLM-030] Aya (`LLME-CASE-007`) demonstrates that multilingual coverage can be made explicit, but its reported model and language results do not transfer to Mosaic Desk. The staged-recipe evidence in `LLME-CASE-008` and the Self-Instruct synthetic-data route remain bounded to their own data and evaluations; generated data can reproduce generator errors. Neither case licenses an aggregate-only conclusion.

Report each declared language and mixed-language segment separately. Preserve comparable task intent where translation is appropriate, but do not assume translation preserves shorthand, cultural context, evidence conventions, or error severity. A language authority reviews ambiguous cases; an LLM engineer cannot declare linguistic adequacy from fluent output.

The failure rule is deliberately asymmetric: improvement in English cannot compensate for a forbidden Gujarati citation regression. Low-volume segments remain visible with their sample sizes and uncertainty. If a segment lacks adequate evidence, record `insufficient-evidence`; do not merge it into an average until it disappears.

Decide with raw behavior and authority intact

Selection begins with hard gates, not rankings. Reject identity mismatch, invalid masking, leakage, unauthorized data access, non-finite optimization, invalid resume, evaluator drift, or a forbidden regression. For survivors, compare target value, uncertainty, retention, language variation, resources, reversibility, and operational consequences.

Data and domain owners validate examples and meaning. Language reviewers own linguistic adequacy. Security and safety authorities own their controls. Platform/MLOps owns reliable training infrastructure and resource accounting. LLM engineering binds the experiment to behavior evidence and recommends retain, stop, or reject. It does not waive another authority's gate.

The managed model remains a dated comparator under the same external contract. Its internal objective and weights are opaque, so compare observable behavior and declared service properties without pretending it was trained under the open-weight recipe.

Complete the simulated SFT decision

`mosaic-sft-checkpoint-simulation.json` retains the real block, exact synthetic identities, illustrative curves, candidate matrices, language slices, hard gates, and decision logic. Its `trainingExecuted` field is false. `sft-early` is only the simulated SFT comparator forwarded to Chapter 11; it is not an artifact and cannot be deployed.

Practice

Audit the token-dominance failure, compare all three rows, and write a checkpoint disposition. Pass only if you reject `sft-late` despite its lower simulated loss and higher aggregate target score, preserve the Gujarati and abstention failures, and state why the exercise produces no training claim.

11. Adapt Efficiently With PEFT

> Evaluate low-rank adapters as traceable behavior-and-resource candidates whose base, tokenizer, template, runtime, quantization, and merge state must remain compatible.

An adapter is a dependency, not a tiny model

Chapter 10 forwarded a simulated SFT comparator, not a trained checkpoint. The real `MD-12` run remains blocked. Chapter 11 adds equally explicit PEFT simulations so we can ask the engineering question: if two candidates meet the behavior contract, what resource and lifecycle evidence distinguishes them?

LoRA freezes the base weights and learns low-rank updates on selected transformations, reducing the trainable-parameter and delta-storage burden in the settings studied by its authors. [CLM-031] `LLME-CASE-009` bounds that adapter evidence to its evaluated settings. The formulation does not promise equal quality, safety, latency, or total cost for a new model and task. Small trainable state still depends on a large base, tokenizer, template, runtime, evaluators, and serving plan.

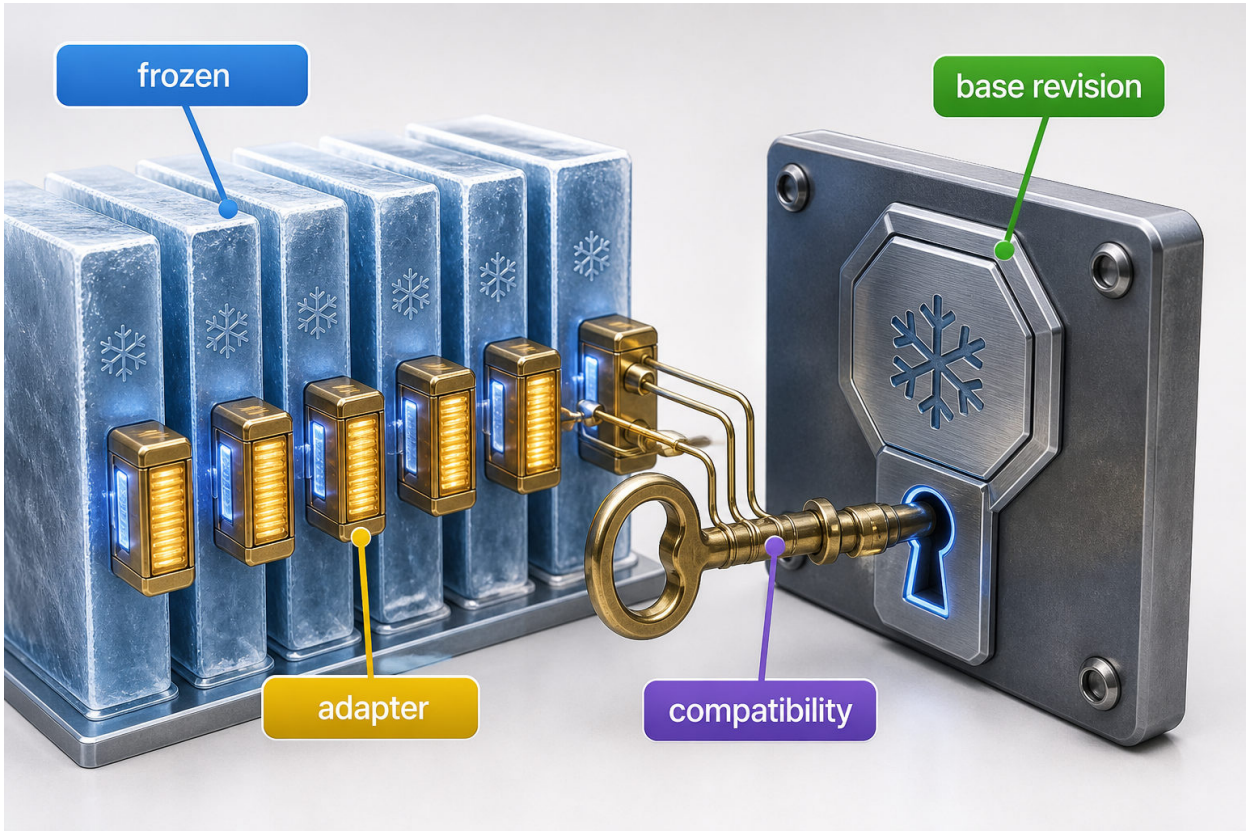
For a transformed weight matrix, the adapter represents an update through lower-dimensional factors. Rank limits the update subspace; scale controls its contribution. Target modules decide where updates enter. Dropout and initialization shape the experiment. These are hypotheses, not magic constants.

Pin the entire compatibility tuple

The Mosaic adapter manifest binds:

- base model ID, revision, configuration digest, and weight digest;
- tokenizer ID, revision, vocabulary, special-token map, and digest;
- training and serving template IDs plus golden serialization;
- runtime, library, dependency lock, precision, quantization, and hardware class;
- adapter method, target modules, rank, scale, dropout, seed, and artifact digest;
- training-data, mask, experiment, evaluator, and checkpoint lineage;
- merged or unmerged state and the exact merge procedure;
- license/provenance facts, limits, owner, and disposition.

The teaching fixture uses the exact base revision `rev-synthetic-a1`, tokenizer `tok-synthetic-a1`, and runtime `1.0.0-teaching`. Because the serving template mismatch remains unresolved, every PEFT row has `executable: false`. Resource and behavior values are deterministic fixtures for comparison, not measurements.



V2-F11.1 - The adapter is small; its dependency surface is not. Essential labels: frozen, adapter, base revision, compatibility. Texture and connector shape supplement color. The figure supports `CLM-031`; it makes no quality claim.

Text alternative: Small trainable low-rank adapter inserts sit beside frozen base-model layers, while a compatibility key binds the adapter to one exact base revision.

Design bounded adapter candidates

Start with a reason for each variable. If the residual concerns a narrow shorthand mapping, attention-projection targets with two bounded ranks may be reasonable hypotheses. If evidence suggests broader representation change, do not silently expand modules; create a new experiment.

The simulation declares two candidates: `peft-r8-attn` and `peft-r16-attn-mlp`. It reports illustrative trainable-parameter counts, peak-memory units, adapter-storage units, training-time units, load time, and merged/unmerged inference consequences. Numbers are comparable only inside this synthetic fixture.

PEFT method support differs across layers, quantized bases, merging, hotswap, and runtime behavior. [CLM-032] Maintained PEFT documentation is the implementation authority for its current interfaces; QLoRA reports resource results for specified hardware, models, quantization, optimizer, and tasks. Recheck library and runtime support at execution time. Never import a paper's memory ratio as a capacity promise.

Quantized-base adaptation adds precision surfaces: quantization format and calibration, compute dtype, dequantization path, optimizer state, kernels, and merge support. A base that loads is not necessarily a base that can train, merge, hotswap, or serve under the intended stack.

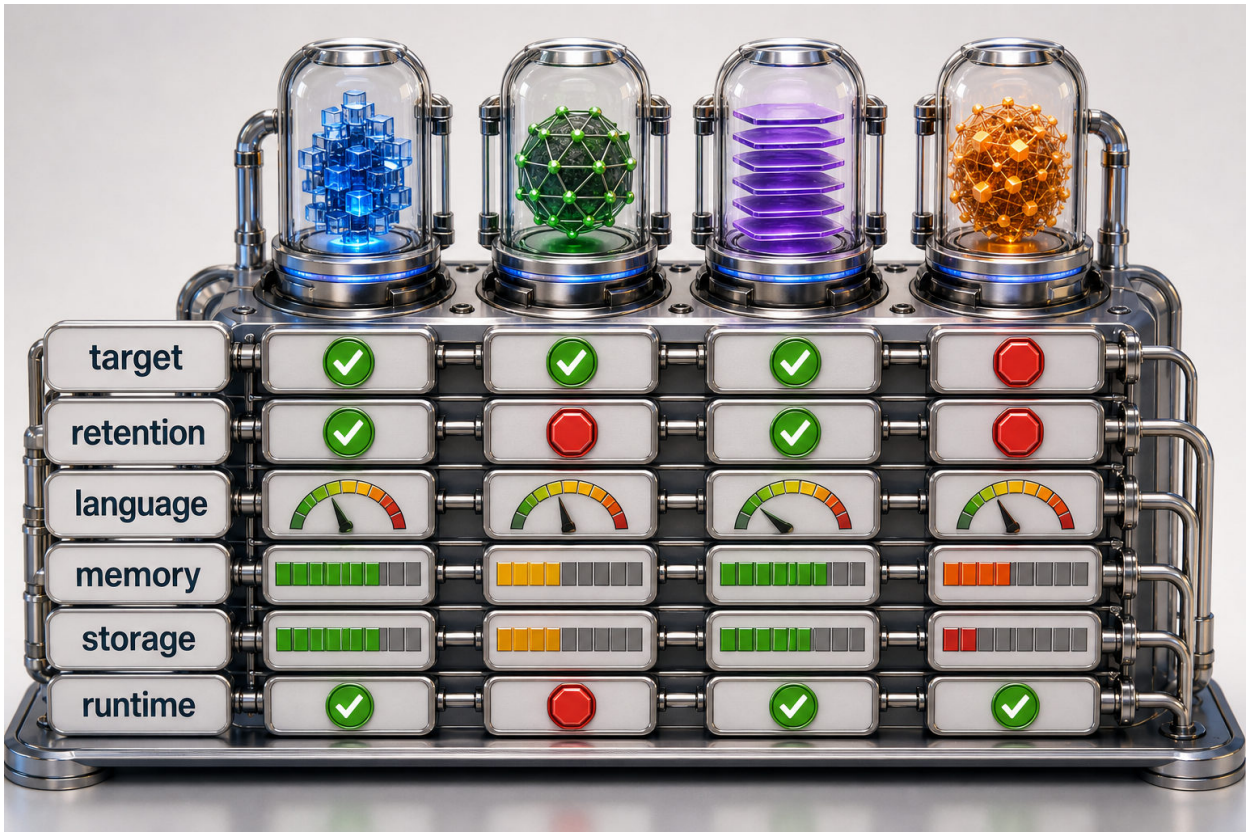
Resource evidence needs definitions. Peak allocated memory differs from reserved memory. Adapter bytes omit the base, tokenizer, runtime, and replicas. Training duration depends on sequence distribution, batching, hardware, topology, checkpointing, and logging. Measure under the exact candidate identity and publish raw definitions.

Test wrong-base and wrong-template failures first

The failure injection pairs an adapter for `rev-synthetic-a1` with `rev-synthetic-a2`. Shapes happen to match, so a permissive loader could accept it. The compatibility validator rejects before load because the base digest differs. Another candidate uses the right base but the wrong tokenizer digest. A third serves without the training template. All fail closed.

Merged and unmerged forms are distinct artifacts. If merging is supported, compare logits or behavior within a declared tolerance under the same runtime and precision. Record the merge algorithm, source adapter, destination base, dtype, digest, and irreversibility assumptions. “Merged successfully” means the operation completed; it does not mean behavior is equivalent.

Hotswapping requires its own support and state-isolation checks. Concurrent adapters must not leak between requests. Caches and batching must carry adapter identity. These serving-platform mechanisms remain platform/MLOps responsibility, while the LLM engineer specifies the behavior and compatibility evidence the platform must expose.



V2-F11.2 - Efficient adaptation lives on a behavior-resource frontier. Essential labels: target, retention, language, memory, storage, runtime. Axis shapes and pass-block symbols supplement color. The figure supports `CLM-032` and `CLM-033`; values are illustrative.

Text alternative: No-tune, supervised, and two adapter candidates are compared across target behavior, retention, language slices, memory, storage, and runtime compatibility rather than parameter count alone.

Compare behavior and resources on one frontier

An efficient adaptation must still pass the same target, retention, control, and inference-compatibility evidence as every other candidate. [CLM-033] Evaluate no-tune, SFT, and PEFT with identical held-out cases, generation policy, raw-output retention, uncertainty, and hard gates.

Order the decision:

1. reject incompatible artifact tuples;
2. reject forbidden target, retention, language, abstention, citation, untrusted-instruction, or no-effect transitions;
3. identify candidates that add target value beyond uncertainty;
4. compare resource, storage, inference, merge, rollback, and operational complexity;
5. retain a Pareto frontier rather than hiding tradeoffs in one score;
6. ask named owners whether any surviving candidate warrants a bounded next step.

The companion's `peft-r8-attn` fixture passes the illustrative protected gates and uses fewer synthetic training-memory and delta-storage units than the SFT fixture. `peft-r16-attn-mlp` raises the aggregate target fixture but fails Gujarati citation retention. It is rejected, regardless of its trainable-parameter efficiency. None is a trained artifact.

The managed model remains the no-weight-access comparator. A managed fine-tuning service, if later considered, must expose enough versioned data, job, checkpoint, evaluation, and rollback identity to satisfy the common behavior contract. We do not invent hidden weight or optimizer details.

Close MD-12 as a simulated decision, not an execution

`mosaic-peft-candidate-simulation.json` records the frozen tuple, LoRA hypotheses, resource definitions, wrong-base failure, four-way matrix, hard gates, and a simulated frontier. `trainingExecuted` and `artifactsCreated` remain false. The actual MD-12 disposition remains blocked; only the teaching comparison marks `peft-r8-attn` as the candidate carried into the Chapter 12 simulation.

Practice

Detect the wrong-base adapter, compare the two PEFT rows with no-tune and SFT, and justify a disposition without using parameter count as quality. Pass when compatibility, target, retention, language, controls, resources, merge state, and authority are all explicit.

12. Optimize Preferences Carefully

> Treat preference optimization as a bounded proxy experiment whose value must survive independent target, retention, language, control, and bias evaluation.

Preference is evidence about a criterion

`MD-12` has not executed training. It carries deterministic no-tune, SFT, and PEFT comparison fixtures into `MD-13`. Chapter 12 asks whether preference optimization would add distinct value to the remaining error categories. It does not assume that a more advanced method deserves a run.

The inherited preference evidence already disclosed a length bias: raters tended to choose longer answers even when the concise answer cited stronger evidence. That is not noise to average away. It is a mechanism by which an optimizer can learn the presentation artifact instead of the intended contract.

In its original formulation, Direct Preference Optimization optimizes a policy relative to reference behavior from preferred and rejected response pairs without separately fitting an explicit reward model. [CLM-034] This removes one component of a particular RLHF pipeline; it does not remove a proxy, reference, data, configuration, or evaluation problem.

For Mosaic, every pair binds task intent, allowed evidence, two response identities, a criterion-specific rubric, randomized order, rater population, disagreement, provenance, and adjudication. The pair cannot simply say “A is better.” Better at evidence linkage, abstention, language adequacy, or concision are different learning signals.

V2-F12.1 - The preference objective remains coupled to data and reference. Essential labels: pairs, policy, reference, objective, candidate, independent eval. Connector direction and gate shape supplement color. The figure supports `CLM-034` and `CLM-035`; it is method-level, not a training result.

> **Text alternative:** Audited preference pairs feed a policy and frozen reference into a preference objective, producing a candidate that must pass an independent evaluation gate.

Freeze a question the proxy can fail

The experiment question is not “does preference accuracy rise?” It is: does a bounded preference intervention improve evidence-linked concise summaries beyond the best no-tune/SFT/PEFT comparator while preserving citation, Gujarati, abstention, untrusted-instruction, and no-effect behavior?

Preregister:

- exact starting policy and frozen reference identities;
- tokenizer, template, runtime, precision, and generation tuple;
- preference-data version, criteria, pair counts, rater and generator coupling;
- objective variant and configuration, including the reference-strength parameter;
- optimization envelope, checkpoints, resources, and stop rules;
- a proxy development measure used only for diagnosis;
- independent held-out target, retention, language, control, and shortcut suites;
- blinded order, judge versions, human/domain adjudication, and disagreement handling;

- what distinct gain would justify the extra method and lifecycle burden.

Preference optimization can overfit biased or shortcut-bearing judgments, so independent behavior and retention evaluation is required. [CLM-035] Hold shortcut cases out of optimization. Include matched concise/verbose answers, swapped positions, style perturbations, confident unsupported details, and cases where abstention is correct. Inspect raw failures instead of presenting only win rate.

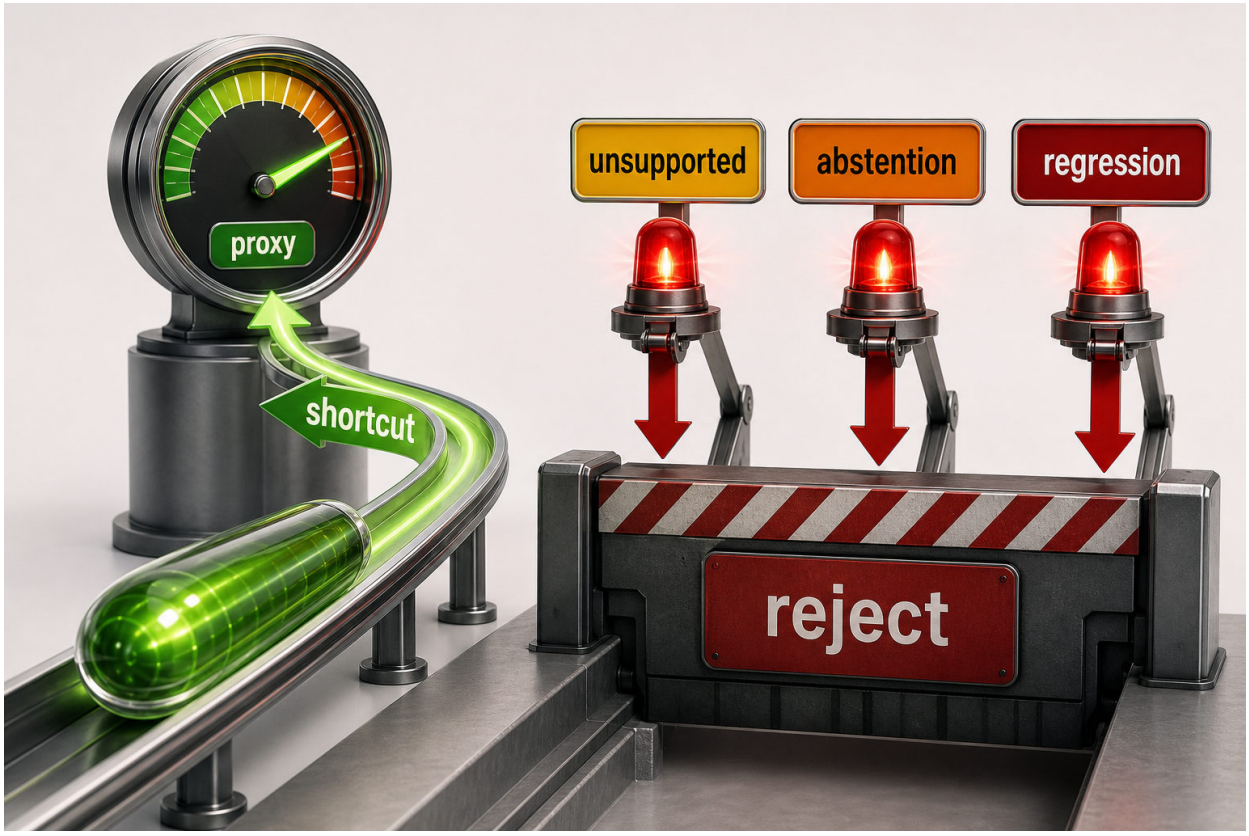
The companion simulates `preference-proxy-candidate`. Its shared grader proxy rises above all comparators. Independent evidence-linked correctness does not improve over `peft-r8-attn`; unsupported-detail and abstention failures increase, especially in Gujarati-English cases. The candidate is rejected as `proxy-only-gain`. These numbers are deterministic fixtures and no optimization ran.

Break the coupling chain

Using the same model family to generate preferences, train, and judge creates coupled evidence that must be disclosed and independently checked. [CLM-036] Independence is graded rather than binary. A different prompt to the same model is not a wholly independent judge. A different model trained on overlapping preferences may share the same bias. Human raters may also share interface or verbosity incentives.

Build an evidence matrix that names the generator, rater, adjudicator, candidate, reference, automated judge, and domain truth source. Mark shared providers, model families, prompts, datasets, annotator pools, and rubric authors. Then add the most independent feasible check for the highest-consequence claims.

Model judges can be useful instruments when calibrated on a versioned sample against qualified human or domain review. Measure position sensitivity, verbosity preference, self/family preference, agreement by criterion and language, and consequential disagreement. A score from an uncalibrated judge is a hypothesis, not approval.



V2-F12.2 - A proxy can reward the visible style and miss the contract. Essential labels: proxy, shortcut, unsupported, abstention, regression, reject. Alarm icons and barrier shapes supplement color. The figure supports `CLM-035` and `CLM-036`; the scene is a failure simulation.

> **Text alternative:** A verbose confident response takes a shortcut to a high proxy score while unsupported-detail, abstention, and language-regression alarms block promotion.

Compare every candidate through the same slices

The Chapter 12 matrix retains four paths: no-tune, simulated SFT, simulated PEFT, and simulated preference optimization. All use the same held-out target, citation retention, neutral-instruction retention, Gujarati, English, Gujarati-English, abstention, conflict, untrusted-instruction, unauthorized-source, and no-effect cases.

Begin with hard gates. A candidate that fails abstention or unsupported-detail rules is rejected before aggregate comparison. For survivors, compare case-level target improvement beyond uncertainty, language spread, resource envelope, compatibility, rollback, and operational complexity. Proxy score is shown in a separate column and cannot satisfy a behavior gate.

Resource evidence includes illustrative training-memory, time, checkpoint/delta storage, reference-policy overhead, evaluation calls, and human adjudication burden. Preference optimization can cost more than its optimizer run: pair production, audits, frozen-reference storage, judge calibration, and disagreement review belong in the ledger. The fixture reports units rather than hardware promises.

The exact compatibility tuple remains `rev-synthetic-a1` plus `tok-synthetic-a1` plus the pinned teaching runtime. Because template compatibility is blocked, the preference candidate is also non-executable. A preference method cannot bypass a failed preflight inherited from SFT or PEFT.

Issue run, stop, or reject without borrowing authority

Run only if the remaining error is preference-sensitive, pair evidence is authorized and criterion-specific, the reference is valid, compatibility passes, independent suites exist, coupling is disclosed, and a distinct-value threshold is preregistered. Stop on identity drift, proxy runaway, loss of judge calibration, forbidden regression, budget breach, or authority withdrawal. Reject when the candidate only improves its optimization proxy or duplicates a cheaper candidate within uncertainty.

Product and domain owners define desired behavior and consequence. Language authorities judge linguistic adequacy. Privacy/legal authorities govern permissible pair and feedback use. Safety/security authorities own relevant controls. LLM engineering implements the objective and evidence system, but a grader score cannot grant release authority.

The DPO paper (`LLME-CASE-010`) reports outcomes for its evaluated tasks and formulation. The MT-Bench and RewardBench evidence (`LLME-CASE-005`) documents bounded judge/reward-model behavior, not universal correctness. Mosaic transfers only the method questions and controls, not their numerical outcomes.

Complete MD-13 with an explicit rejection

`mosaic-preference-optimization-simulation.json` records the frozen policy/reference tuple, pair audit, coupling map, proxy and independent matrices, resources, hard gates, and an explicit rejection. `trainingExecuted` is false and `artifactsCreated` is false. The remaining error ledger, not the rejected preference score, will guide Chapter 13's method decision.

Practice

Blind and swap the shortcut pairs, audit the coupling map, compare all four candidates, and issue run, stop, or reject. Pass only if independent target and protected behavior outweigh the proxy increase and the named authorities remain explicit.

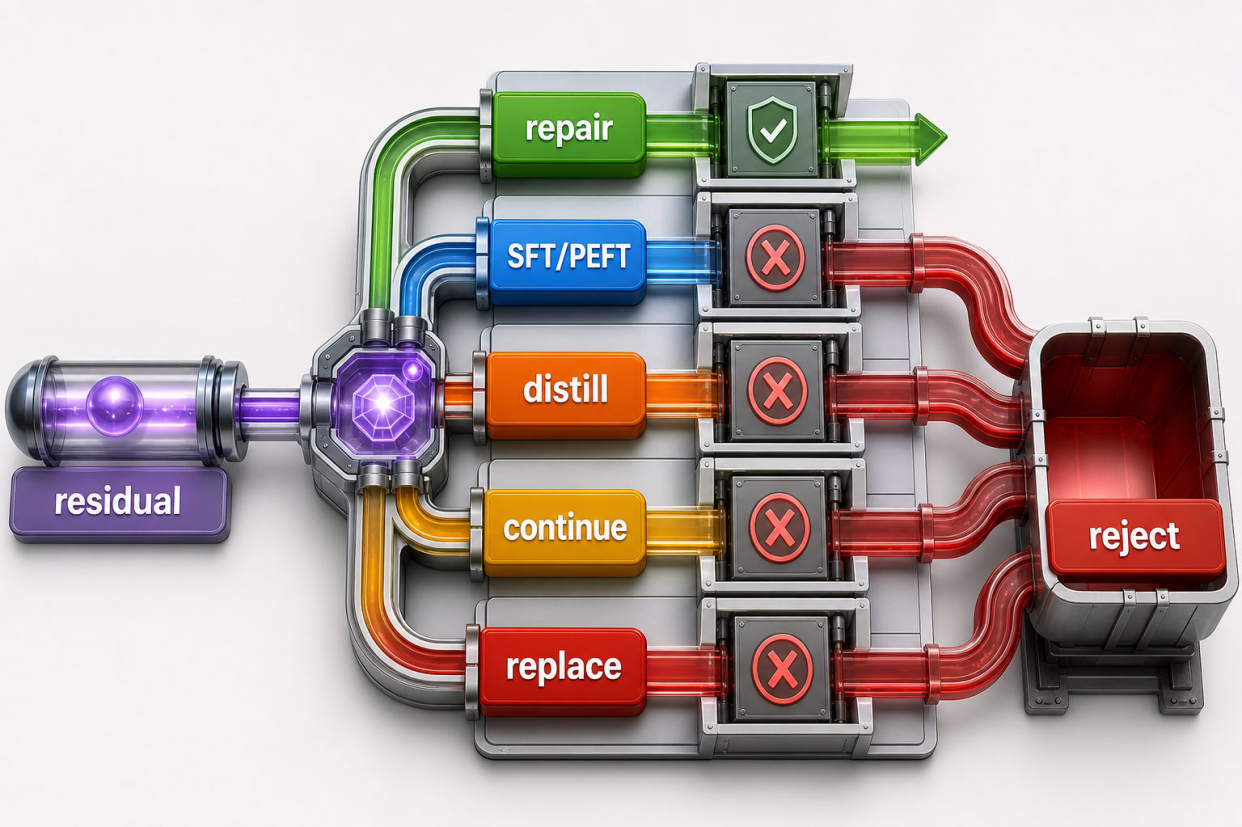
13. Decide on Distillation or Continued Pretraining

Choose advanced adaptation only when its mechanism, transfer source, authority, and disconfirming evidence match a distinct residual gap.

Escalation needs a distinct thesis

MD-13 opens with the accepted preference rejection. The remaining ledger names bounded shorthand coverage, unresolved template compatibility, and insufficient independent language evidence. None automatically implies distillation or continued pretraining.

Begin with alternatives: keep no-tune, repair prompt/context/retrieval, use bounded SFT or PEFT, replace the base or managed comparator, or stop. Ask whether the gap is stable behavior, missing distribution, capacity, or changing knowledge; what smaller path failed; what teacher or corpus carries the signal; and what held-out evidence could disconfirm the method.



V2-F13.1 - Methods answer different failure hypotheses. Essential labels: residual, repair, SFT/PEFT, distill, continue, replace, reject. Branch shapes supplement color. Supports CLM-037 to CLM-039; not a ranking.

Text alternative: Residual diagnosis routes separately to repair, SFT or PEFT, distillation, continued pretraining, replacement, no project, or rejection.

Distillation transfers teacher behavior and defects

Distillation transfers teacher behavior into a student under an explicit objective and data process; it is not generic compression magic. [CLM-037] Teacher distributions, generated targets, or demonstrations remain bounded by teacher quality, student capacity, data, decoding, objective, and evaluation.

A teacher ledger pins model or service version, prompt, generation settings, inputs, filtering, rights, cost, unavailable internals, and limits. Targets that make grounded claims need authorized evidence links. Mosaic's failure injection produces fluent policy targets containing unsupported details. They are rejected before a pilot; grammar filtering would preserve the transferred defect.

Distillation is plausible for a stable bounded behavior when a smaller student must meet a resource goal and independent suites can detect lost capability. It is mismatched for changing knowledge, broken templates, or missing authorization. A bounded pilot would cap calls, updates, data, time, storage, and evaluation, then compare teacher, student baseline, distilled candidate, and the existing best path. Teacher agreement is not acceptance.

Continued pretraining changes a distribution

Continued pretraining targets a domain or task distribution through additional unsupervised language modeling and carries contamination and retention costs. [CLM-038] It may fit a demonstrated language or domain representation gap with an authorized corpus. It does not fit frequently changing policy that belongs in retrieval.

A corpus ledger records sources, permission facts for legal interpretation, dates, language/domain mixture, transformations, deduplication, restricted-data review, contamination, split relationships, hashes, retention, and omissions. Dolma (`LLME-CASE-006`) and domain-adaptive pretraining research provide bounded process evidence, not Mosaic authorization or forecasts.

The failure injection proposes memorizing volatile policy. Mosaic rejects it: weights obscure freshness and correction while retrieval already owns current authority. Independent target, retention, language, abstention, citation, control, and no-change suites remain required because corpus loss cannot select product behavior.

V2-F13.2 - Transfer sources remain bounded by lineage and tests. Essential labels: teacher, corpus, provenance, rights, contamination, evaluate. Supports CLM-037 and CLM-038.

Text alternative: Teacher and corpus paths cross provenance, rights, contamination, and independent evaluation barriers before reaching a student or base model.

Reject fashion with common evidence

Advanced adaptation should be rejected when its hypothesis, data, resource, or evaluation burden is not distinct from a smaller intervention. [CLM-039] Scaling and staged-recipe research describes studied regimes; it cannot forecast Mosaic fitness from parameters, tokens, or method names.

Compare residual mechanism, transfer authority, target/protected evidence, resources, compatibility, rollback, change cadence, uncertainty, and owners on one sheet. Research owns novel methods; legal/privacy/domain authorities approve transfer sources; platform owns compute; LLM engineering owns method fit and behavior evidence.

The deterministic companion rejects continued pretraining because policy is volatile and rejects distillation because teacher targets lack evidence and add no distinct value. No pilot runs. Chapter 14 receives the simulated peft-r8-attn lineage with an explicit non-executable state.

Design a pilot that can stop

If a future residual clears the method gate, the pilot record freezes the question before computation. For distillation it names the teacher and student identities, target-generation route, temperature or decoding policy, evidence validator, rejected-target archive, student objective, and teacher-call budget. For continued pretraining it names corpus identity, sampling mixture, tokenizer, sequence policy, objective, update budget, checkpoints, and contamination scan. Both routes inherit the exact baseline, protected suites, and package compatibility tuple.

Each pilot has automatic stops: unauthorized source access, identity or template mismatch, contaminated holdout, non-finite optimization, resource breach, invalid checkpoint, forbidden retention transition, target futility, or authority withdrawal. A stop is evidence about the thesis; it is not a reason to widen scope mid-run.

Resource comparison includes data acquisition and review, teacher generation, training memory and time, evaluation, storage, inference consequences, human adjudication, and maintenance. Distillation that reduces inference cost may add recurring teacher and refresh costs. Continued pretraining may create a larger artifact and more expensive future adaptation. Report the whole lifecycle rather than only training parameters.

Preserve the no-project option

The decision sheet records what happens if Mosaic does nothing. The current no-tune or simulated PEFT path may remain preferable when evidence is insufficient, the gap is low value, or the authority and maintenance burden outweigh likely benefit. “No project” is not failure; it is a controlled disposition with revisit triggers.

Revisit only when a stable residual appears across fresh held-out cases, a suitable authorized teacher or corpus exists, compatibility is repaired, and a bounded pilot can distinguish the method from simpler alternatives. This prevents repeated exploration from quietly becoming an indefinite training program.

The Tulu 3 case (`LLME-CASE-008`) demonstrates why stage-level artifacts and comparisons matter, but its recipe does not select Mosaic's stages. The corpus case demonstrates process and contamination concerns, not blanket permission. Every imported lesson remains attached to its model, data, task, and measurement limits.

Practice

Classify a latency-constrained stable behavior, changing policy, and language-distribution deficit. Pass only when mechanism, authority, contamination, retention, resources, and disconfirmation justify pilot or rejection.

Part IV - Engineer the Runtime Envelope

A model file is not a model system. Its behavior depends on tokenizer, template, adapter, schema, defaults, precision, quantization, runtime, routing, and surrounding controls. A variant can pass an integrity check and still be incompatible. A fast request average can hide context growth, queue amplification, memory pressure, decode tails, or protected behavior regression.

Part IV makes the deployment candidate and its runtime envelope explicit. Chapter 14 binds weights, tokenizer, template, adapter, configuration, schema, licenses, provenance, checksums, compatibility, promotion state, and rollback into one fail-closed release identity. Chapter 15 separates weight memory, KV state, prefill, decode, batch, concurrency, queue, throughput, and tails in a workload-specific resource model and names the specialist handoffs it cannot resolve. Chapter 16 compares precision, quantization, batching, caching, routing, and degradation candidates only after replaying target, retention, language, abstention, control, and compatibility gates.

Mosaic Desk advances through `MD-14` and `MD-15`: an incomplete package inventory, an inference-capacity simulation, and a synthetic serving frontier. Integrity alone cannot promote the package because no trained adapter exists and template compatibility remains unresolved. Resource values illustrate calculations and transitions; they are not accelerator benchmarks, capacity promises, or cost forecasts.

The handoff into Part V is a reconstructable but blocked release candidate with an explicit frontier and fallback ladder. The final part must show how evidence, exposure, signals, diagnosis, rollback, migration, and authority behave when the release decision is hold rather than inventing success for narrative closure.

14. Package and Version the Model System

Bind every behavior-facing component, lineage link, compatibility result, limitation, and rollback state into one fail-closed release identity.

A checkpoint is not a model system

MD-14 receives simulated peft-r8-attn only as a packaging exercise. No adapter bytes exist and template compatibility remains blocked. A trustworthy manifest must preserve that state.

Deployable identity must bind base or checkpoint, tokenizer and template, adapter, configuration, schema, runtime, provenance and license facts, and integrity metadata. [CLM-040] Inventory base revision and digest; tokenizer and special tokens; exact template and golden serialization; adapter configuration and merge state; generation defaults and schema; dtype, quantization, runtime, hardware and lock; data/training/evaluation lineage; rights facts, limitations, owners, promotion and rollback.

Use content identities, not final-v2 filenames. Every digest names algorithm and object scope. Missing artifacts remain absent and never receive pretend hashes.

V2-F14.1 - Release identity spans the behavior-facing system. Essential labels: base, tokenizer, template, adapter, config, schema, runtime, license, evidence, hash. Supports CLM-040 and CLM-041.

Text alternative: One versioned crate contains separately labeled base, tokenizer, template, adapter, configuration, schema, runtime, license, evidence, and hash compartments.

Trace claims to sources

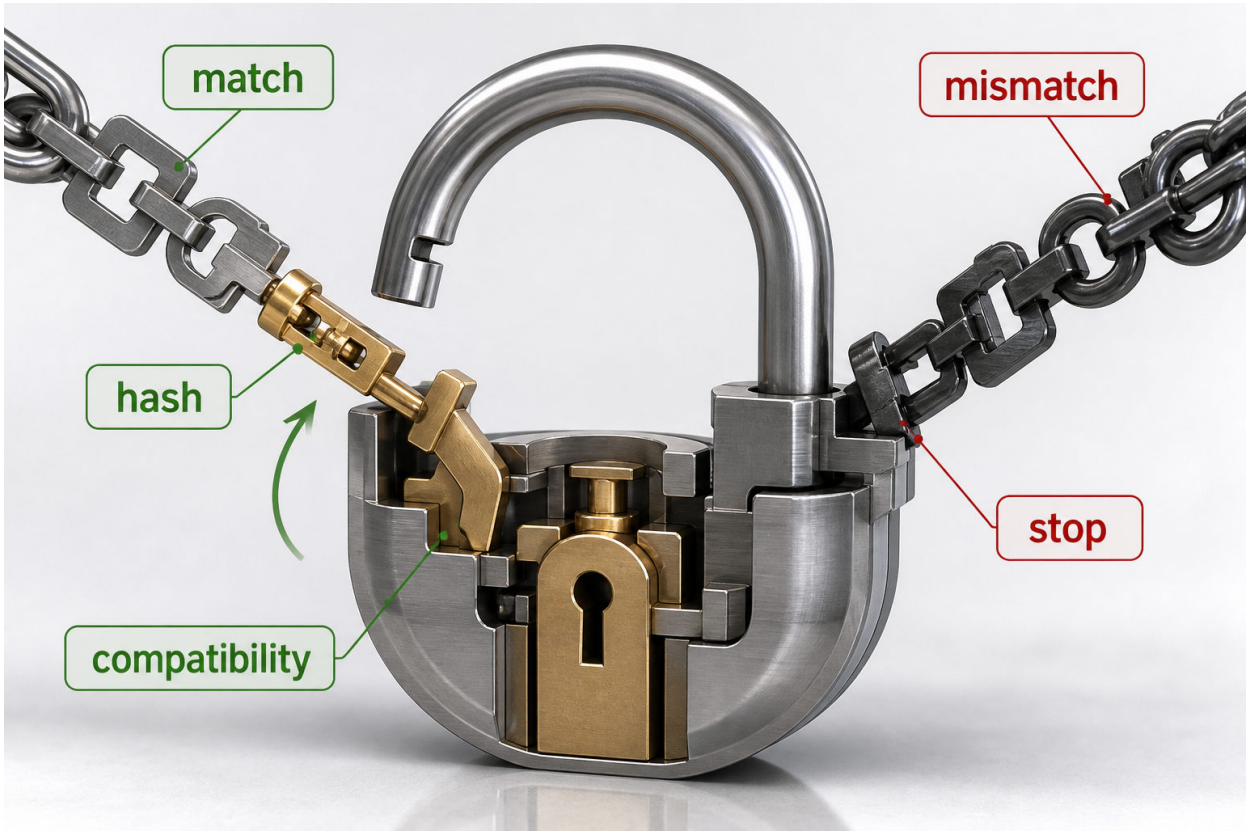
Lineage runs from MD-09 hypothesis and inspection through MD-10 data, MD-11 learning evidence, MD-12 plan/simulations, and MD-13 rejection. It answers which records could influence a candidate, which renderer/mask would be used, which run produced it, and which evidence permits promotion. Here the run is none, so package status is blocked-plan-only.

Managed dependencies retain provider, dated endpoint/model ID, API/schema, region posture, options, evaluation snapshot, and fallback. Hidden tokenizer, weights, kernels, and topology are not invented.

Verify combinations before load

Compatibility checks base/config digests, tokenizer/embedding assumptions, special-token behavior, template family and golden serialization, adapter base/targets/rank/dtype/quantization, merge state, schema/stops/defaults, and runtime support.

The failure injection supplies a validly hashed adapter with the wrong tokenizer and old template. Integrity passes; compatibility stops before load. A second fixture has the right tuple but no adapter artifact. It remains incomplete.



V2-F14.2 - Integrity cannot substitute for compatibility. Essential labels: match, mismatch, hash, compatibility, stop. Supports CLM-042.

Text alternative: Matched base, tokenizer, template, adapter, and runtime keys open a lock while a validly hashed mismatched combination stops.

Cards disclose; authorities decide

Model cards document intended use, evaluation, and limitations but do not approve deployment. [CLM-041] Mosaic's card states the fictional evidence, non-executable package, template failure, protected suites, rejected candidates, excluded uses, untested segments, maintenance triggers, and required authorities.

Legal interprets licenses. Security owns supply-chain controls. Platform owns registries and promotion. Product, domain, and language authorities own fit and consequence. Packaging completeness grants none of these approvals.

Artifact format and checksums improve integrity and handling but do not prove provenance or behavioral compatibility. [CLM-042] Safe serialization can reduce executable deserialization surfaces; it cannot make malicious, mislicensed, or unsuitable weights trustworthy. Verify provenance and digests at acquisition, storage, promotion, and load, then replay behavior fixtures.

Rollback restores the prior complete package: weights, tokenizer, template, schema, defaults, runtime binding, and routing. The companion inventories components, binds lineage, rejects three mismatches, and sets promotion to blocked-missing-artifact-and-template-compatibility.

Make promotion a state machine

A package moves through explicit states: inventory-incomplete, integrity-verified, compatibility-verified, behavior-evidence-complete, authority-reviewed, release-candidate, staged, and released. Skipping a state is invalid. A failed check returns to a named earlier state or rejects the package; it does not receive a verbal exception hidden in notes.

For Mosaic, base and tokenizer digests are synthetic teaching identities, the adapter is absent, and training-versus-serving templates conflict. The state machine therefore stops at inventory-incomplete and compatibility-blocked. It cannot be called a release candidate even though the JSON schema validates.

Compatibility fixtures include golden serialization of a normal dialogue, system control, tool-shaped data, empty/long inputs, Gujarati-English shorthand, stop-token behavior, padding, and structured output. Replay compares exact bytes where deterministic and semantic contract outcomes where runtime variation is expected. Tolerances must be declared before comparison.

The adapter-lineage case (`LLME-CASE-009`) is bounded: LoRA and PEFT research motivates binding adapter and base, but no paper proves this fictional package. The lifecycle case (`LLME-CASE-013`) shows that managed APIs and schemas change; it supports dated identities and migration evidence, not permanence.

Write a useful change card

The change card answers what changed, why, from which evidence, who is affected, which cases improved or regressed, what resources changed, which limits remain, and how to detect and recover from failure. It links raw cases and rejected candidates rather than presenting only a winning score.

Known limitations are actionable: template incompatibility blocks execution; no adapter bytes or training run exist; language evidence is finite and synthetic; provider internals are unavailable; safety, legal, privacy, domain, and production approval are absent. Each limitation names an owner or revisit condition without implying that the LLM engineer owns every resolution.

Rollback verification checks that the prior package can still resolve, load in its compatible environment, replay critical fixtures, and reclaim routing without ambiguous mixed state. Registry implementation, signing infrastructure, access policy, and disaster recovery remain platform/security depth; the package declares the evidence contract those systems must preserve.

Practice

Verify the manifest and diagnose wrong base, wrong tokenizer/template, and missing adapter. Pass when integrity, provenance, compatibility, behavior evidence, approval, and rollback stay separate.

15. Reason About Inference Memory and Throughput

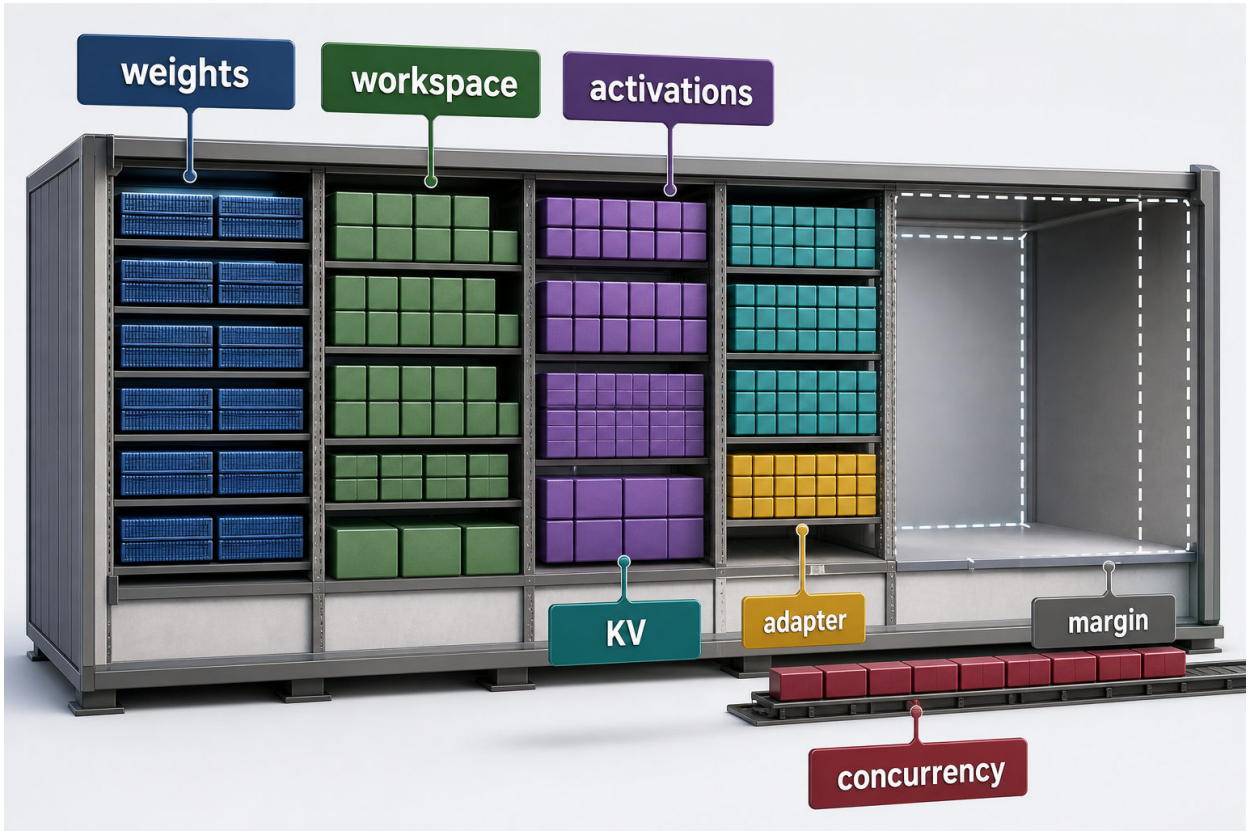
Build a workload-specific resource model that separates weights, KV state, prefill, decode, concurrency, throughput, and latency tails.

Capacity starts with a workload

MD-15 receives a blocked package, so Mosaic builds a deterministic worksheet and synthetic queue trace only. Define short-message, long-thread, and burst classes; input/output distributions; arrival and concurrency; streaming, TTFT, inter-token and end-to-end budgets; cache/privacy rules; errors; and behavior constraints.

Inference memory includes weights plus runtime state such as KV cache whose size depends on model, sequence, precision, and batch or concurrency. [CLM-043] Add activations, workspaces, allocator fragmentation, graph/communication buffers, adapter state, and safety margin.

For a simplified decoder, KV bytes depend on active sequences, retained tokens, layers, key/value factor, KV heads, head dimension, and bytes per element. Architecture, grouping, paging, quantization, and prefix reuse change the result. Expose every factor and validate peak memory on the exact stack.



V2-F15.1 - Weights share memory with growing request state. Essential labels: weights, workspace, activations, KV, adapter, margin, concurrency. Supports CLM-043 and CLM-045.

Text alternative: Finite shelves hold weights, workspace, activations, KV cache, adapter state, and safety margin while concurrent sequences consume cache blocks.

Separate prefill, decode, and user-visible tails

Prefill processes context and creates request state; decode repeatedly produces tokens while reading growing state. Time to first token exposes queueing and prefill. Inter-token latency exposes decode cadence. End-to-end latency also depends on output length.

Prefill and decode have different compute and memory behavior, and batching, caching, and routing alter throughput and tail latency. [CLM-044] Report prompt/output throughput, completed requests, TTFT, inter-token, end-to-end latency, cancellations, timeouts, and errors separately. Preserve p50, p95, and p99 per workload class with sample counts; averages hide bursts.

Warm and cold runs answer different questions. Record load, compilation, cache, allocator, repeated prompts, package, hardware, runtime, and order.

Batching can win throughput and lose interaction

Waiting to batch raises queue delay even when utilization improves. Paged KV, continuous batching, chunked prefill, and prefix caching change the frontier under particular stacks. Published PagedAttention throughput (`LLME-CASE-011`) is not portable.

The fixture compares interactive-small-batch and throughput-large-batch. The larger batch completes more synthetic tokens per unit but breaches p99 TTFT during bursts, so it is rejected for interactive routing.

V2-F15.2 - Utilization can lengthen a user-visible tail. Essential labels: arrivals, batch, cache, TTFT, throughput, p99 tail, limit. Supports CLM-044; trace is synthetic.

Text alternative: Short, long, and burst arrivals enter batching and cache scheduling; a larger batch raises throughput while p99 TTFT crosses the interactive limit.

Keep performance evidence causal and bounded

Hold package/workload fixed while varying one choice. Sweep concurrency and capture queue, prefill, decode, post-processing, peak memory, cache occupancy, batch composition, token counts, cancellations, and failures.

Task-level capacity models should record hardware, runtime, and configuration and escalate kernel or platform work to adjacent specialists. [CLM-045] Precision, kernels, sharding, parallelism, topology, scheduler, and compiler behavior require target-hardware validation. FlashAttention and FSDP explain mechanisms, not universal multipliers.

LLM engineering owns workload definitions, behavior constraints, package identity, and user-visible interpretation. Platform/SRE owns fleet capacity and reliability; kernel specialists own low-level

optimization. Managed paths expose client-observed behavior and declared limits, not hidden hardware or schedulers.

The companion records an explicit memory formula, three workloads, two synthetic traces, percentiles, hard budgets, and specialist handoff. modelExecuted and benchmarkExecuted remain false.

Turn budgets into a load plan

The load plan crosses three request classes with warm and cold state, low-to-burst arrival rates, and bounded concurrency steps. Each cell records scheduled and completed requests, prompt/output tokens, queue time, prefill time, per-token decode intervals, end-to-end duration, memory high-water mark, cache occupancy, batch sizes, cancellation, timeout, and error reason. Raw traces retain package and configuration identity without retaining sensitive prompts.

Begin below expected capacity, then raise one dimension at a time until a declared limit, not until the server crashes. Stop on out-of-memory, invalid output, behavior drift, timeout budget, unstable tail, or measurement corruption. Recovery between cells restores a known warm or cold state. Capacity is the region that repeatedly satisfies all applicable constraints, not the single highest completed sample.

Use Little's Law only when its assumptions and units fit the observed stable interval; do not paste queueing formulas over bursty or overloaded traces. Report offered load separately from completed throughput. A system that drops requests can appear fast if only successful completions are timed.

Keep quality coupled to serving choices

Truncation, prefix reuse, quantization, speculative decoding, batching, and parallel execution can alter behavior or its evidence. The benchmark therefore replays frozen target, citation, abstention, language, schema, and control cases for each serving candidate. A capacity winner that changes contract behavior is rejected before cost comparison.

Speculative decoding evidence is bounded to draft/target pairing and acceptance behavior. Quantization support is bounded to the exact model, layers, calibration, kernels, hardware, and runtime. vLLM and TensorRT-LLM documentation describe fast-moving implementations. Record versions and remeasure instead of importing a headline.

Produce a specialist-ready handoff

When p99 TTFT or memory fails, hand platform and kernel specialists the smallest reproducible package identity, workload generator, trace, hardware/runtime configuration, expected contract, failure threshold, and one-variable experiment. Do not prescribe a kernel rewrite from an application trace. Specialists return measured alternatives and new constraints for behavior replay.

For the managed comparator, the same load distribution and client timing boundaries apply, but internal memory and scheduling remain unknown. Compare observable service behavior, quotas, errors, cost, and change controls without pretending equivalence to open-weight internals.

Practice

Calculate the KV ledger and reject the headline throughput configuration that violates p99 TTFT. Pass when workload, package, hardware, runtime, warm state, tails, errors, behavior constraints, and authority are visible.

16. Compress, Serve, and Preserve Behavior

Select serving variants on an expiring behavior-resource frontier, with exact compatibility, protected replay, and graceful degradation.

Every serving change creates a candidate

MD-15 contains a workload plan, not a benchmark result. The package remains blocked because no adapter exists and the training/serving templates conflict. Chapter 16 therefore performs a deterministic bake-off with synthetic measurements only. modelExecuted, benchmarkExecuted, and artifactCreated remain false.

Freeze the reference before changing anything: base, tokenizer, template, adapter/merge state, schema, generation defaults, precision, quantization, runtime, kernels, hardware class, scheduler, workload, evaluator versions, and raw protected cases. A change to any field creates a new candidate identity.

Candidates may vary weight or cache quantization, calibration data, kernel/runtime, batching, cache policy, speculative decoding, or routing. Change one surface at a time when causal attribution matters. A compound variant needs an ablation plan or it stays an opaque bundle.

Quantization can reduce memory or compute cost, but quality and speed effects depend on method, model, calibration, hardware, and runtime. [CLM-046] Nominal bit width does not determine loaded bytes, kernel support, latency, or task behavior. Scales, metadata, unquantized layers, adapters, cache precision, conversion format, and runtime copies remain in the ledger.

AWQ, GPTQ, QLoRA, bitsandbytes, and TensorRT-LLM describe bounded methods or implementations. Their published perplexity, memory, or speed results do not establish Mosaic behavior. Calibration examples must be authorized, representative of the intended activation distribution, separate from evaluation, and versioned.

`LLME-CASE-012` contributes only the lesson that quantization needs method-specific calibration and behavior replay. Its reported model, benchmark, hardware, and runtime results are non-portable. `LLME-CASE-011` contributes only the serving lesson that KV allocation and scheduling affect a measured workload; its throughput is not a Mosaic capacity estimate.

V2-F16.1 - No single configuration dominates every dimension. Essential labels: behavior, p99 tail, throughput, memory, cost, reject. Marker shapes supplement color. Supports CLM-046 to CLM-048; all points are synthetic.

Text alternative: Reference, quantized, alternate-runtime, and scheduling candidates occupy different positions across protected behavior, p99 tail latency, throughput, memory, and cost, with unsafe points rejected.

Verify compatibility before comparing speed

Each candidate must pass the MD-14 identity gate: base and configuration digests, tokenizer vocabulary and special tokens, template golden serialization, adapter binding, dtype and quantization format, runtime support, schema, stops, and generation defaults. Unsupported kernels do not silently fall back; the observed kernel path and fallback state are recorded.

The first failure candidate reports lower weight-memory units but uses an unsupported kernel and therefore gains no synthetic speed. It is not promoted on memory alone. The second doubles synthetic throughput but increases citation-schema failures on long Gujarati-English inputs. Aggregate target score looks stable because short English cases dominate. The protected slice rejects it.

Replay target shorthand, citation retention, neutral instruction, English, Gujarati, Gujarati-English, abstention, conflict, untrusted instruction, unauthorized source, no external effect, schema validity, and long-context cases. Preserve raw outputs and pass-to-fail transitions. Perplexity and generic benchmark scores remain diagnostic instruments, not acceptance evidence.

Paged KV management, batching, caching, and speculative decoding can improve serving efficiency under bounded workloads. [CLM-047] Their benefits may reverse with model pairing, acceptance rate, sequence

distribution, concurrency, hardware, or implementation. Speculative decoding must preserve the intended distribution under the exact algorithmic assumptions and still pass application behavior.

Measure prompt/output throughput, request throughput, TTFT, inter-token latency, end-to-end p50/p95/p99, errors, memory high-water mark, cache occupancy, batch composition, and cost units. Separate offered load from completed work. Report warm/cold state and sample counts.

Build the frontier after hard gates

Apply compatibility and protected-behavior gates before resource ranking. A candidate with a forbidden language, citation, abstention, schema, or control transition is not on the feasible frontier. Among survivors, remove dominated candidates: another candidate is at least as good on all decision dimensions and better on one.

Serving configuration must be selected on a measured behavior-resource frontier including protected segments and latency tails. [CLM-048] A frontier expires when package, workload, hardware, runtime, traffic, evaluator, or behavior contract changes. It is evidence for one decision, not a permanent leaderboard.

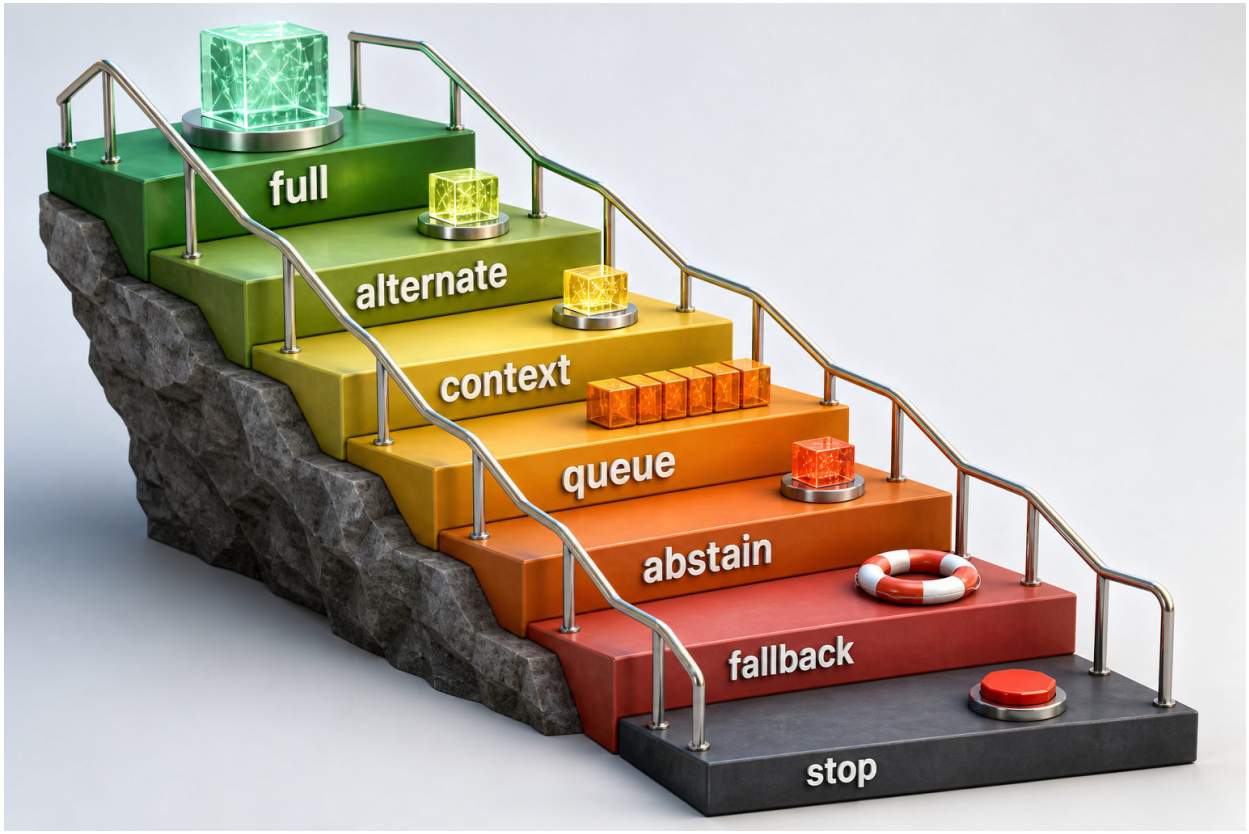
The deterministic companion compares reference-full, quant-fast-unsafe, quant-memory-only, and alternate-runtime-balanced. Reference-full remains the fallback. The unsafe fast point fails Gujarati citation/schema. The memory-only point is dominated because its unsupported kernel provides no latency or throughput gain. The balanced point is only a simulated frontier candidate; the blocked package prevents release.

Degrade deliberately under pressure

Graceful degradation is a predeclared state machine, not arbitrary quality loss:

1. use the verified full reference while budgets pass;
2. use a behavior-qualified alternate candidate only if its identity matches;
3. reduce optional context within contract and expose that state;
4. queue or rate-limit with an honest delay;
5. abstain when evidence, schema, or latency validity cannot be preserved;
6. fall back to the prior managed/no-tune path;
7. stop when no qualified path remains.

Never silently drop authoritative evidence, truncate required citations, switch language handling, disable controls, or route to an unqualified model. A fallback response states its limitations without leaking system details. Recovery requires replaying the affected cases before leaving degradation.



V2-F16.2 - Pressure changes service state, not hidden truthfulness. Essential labels: full, alternate, context, queue, abstain, fallback, stop. Step shapes supplement color. Supports the operational fallback evidence for CLM-048.

Text alternative: Runtime pressure follows bounded steps from full qualified service to alternate candidate, reduced optional context, queue, abstain, fallback, and stop.

Platform selects deployment, capacity, autoscaling, and operational routing. LLM engineering supplies the behavior-resource evidence and degradation contract. Product/domain/language, privacy, security, and release authorities retain their decisions. Managed optimization exposes only provider features and measured external behavior; hidden kernels and hardware are not invented.

MD-15 closes with a deterministic frontier, rejection ledger, and degradation plan. No candidate is released.

Practice

Reconstruct the frontier, reject the fast slice-regressing candidate and the dominated memory-only candidate, then walk a burst through the degradation ladder. Pass when compatibility, target, retention, language, controls, service tails, and fallback evidence determine every transition.

Part V - Operate Adapted Models Through Change

Adapted-model lifecycle work begins before exposure and continues after a rollback. Checkpoints, tokenizers, templates, runtimes, routers, data populations, and evaluators can drift independently. An incident symptom can originate in behavior, retrieval, validation, runtime, capacity, product, or observation. A team that labels every regression "the model" cannot choose a safe correction.

Chapter 17 integrates the complete series evidence chain. The readiness packet reconstructs `MD-01` through `MD-16`, names the missing artifact and compatibility blockers, and issues hold. A tabletop tokenizer/runtime incident preserves privacy-minimized signals, layered hypotheses, containment, rollback, reconciliation, requalification, and change replay. The resulting durable learning updates evaluation, controls, model-system records, monitoring, and ownership without granting the LLM Engineer release or risk authority.

Mosaic Desk completes `MD-16`: a release-hold record, synthetic cohort plan, incident/change replay, and portfolio review. No trained model, live cohort, measured capacity, production incident, or external effect is claimed. The hold is a successful evidence outcome because it prevents an incomplete package from acquiring false release status.

The series closes with a reusable discipline: change the smallest plausible surface, preserve every identity that makes evidence comparable, protect behavior that must not regress, qualify runtime under the real workload, and keep formal authority separate from engineering recommendation.

17. Release, Diagnose, and Evolve Adapted Models

Close the series with an authority-preserving release, incident, rollback, requalification, and durable-learning loop across the complete model-system dossier.

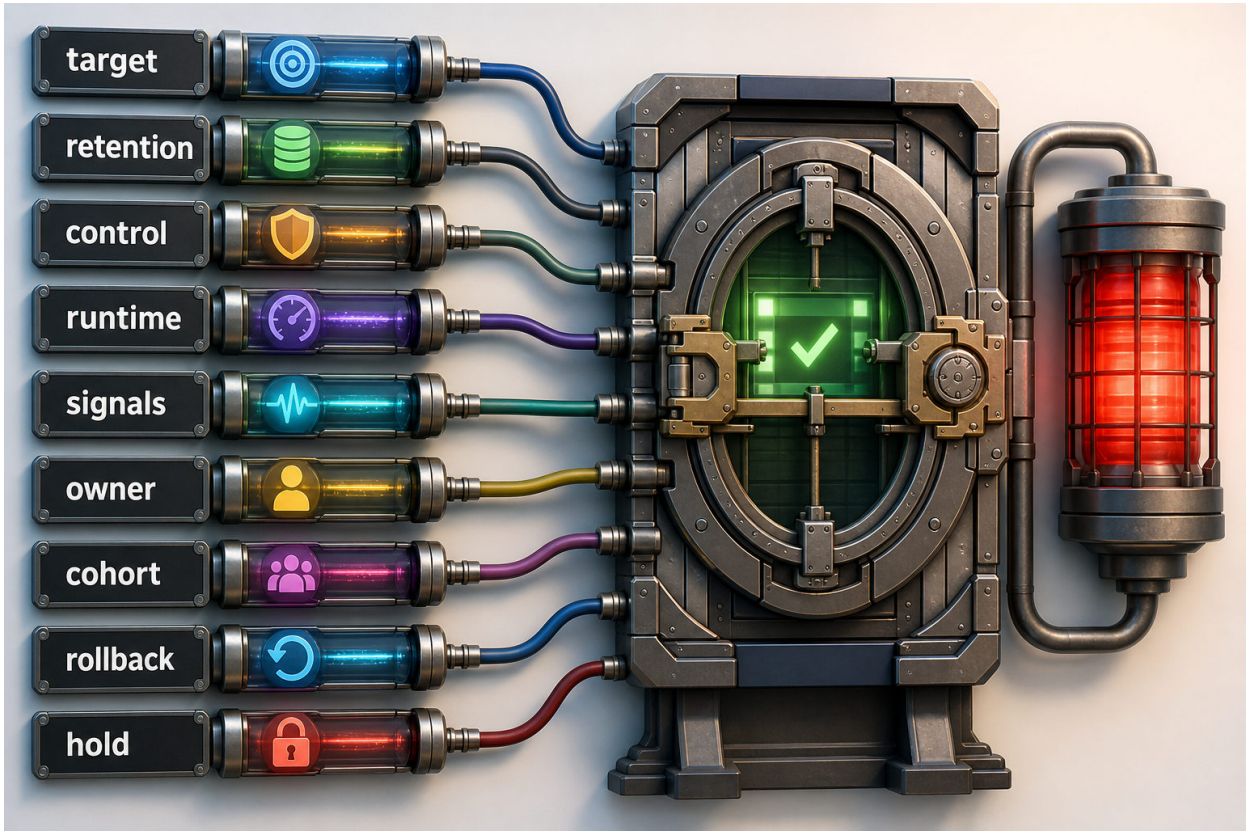
A complete dossier can still say hold

MD-16 consumes every Mosaic artifact from the language-task contract through the simulated serving frontier. The evidence chain is complete enough to reconstruct, but the open-weight package is not executable: the adapter was never trained and the template remains incompatible. The readiness disposition is therefore hold. This is a successful evidence process without a fabricated release.

Adapted-model release needs target, retention, control, runtime, owner, observability, cohort, and rollback evidence. [CLM-049] The packet links exact package identity, data/training lineage, compatibility, behavior/resource frontier, raw protected cases, limitations, privacy review, named decision owners, staged cohort, stop triggers, prior recoverable package, and reconciliation plan.

Offline pass is necessary but not sufficient. The state machine is offline-qualified, shadow, internal, restricted cohort, wider cohort, release, hold, rollback, or deprecate. Each transition names entry evidence, observation window, stop rules, decision owner, and fallback. No deadline changes the evidence label.

Because the package is blocked, Mosaic performs a tabletop only. No user traffic, model call, deployment, or external effect occurs.



V2-F17.1 - Evidence and distributed authority meet at the cohort gate. Essential labels: target, retention, control, runtime, signals, owner, cohort, rollback, hold. Distinct keys supplement color. Supports CLM-049 and CLM-051.

Text alternative: Target, retention, control, runtime, privacy-minimized observability, owner decisions, cohort plan, and rollback evidence enter a release gate that can issue hold.

Observe decisions without collecting the product

Monitoring begins with decisions: what signal triggers contain, diagnose, rollback, or review? Record request class, package and configuration identity, language/slice, schema state, evidence/citation status, abstention, fallback, latency stage, resource state, error code, and cohort. Prefer counts, bounded categories, hashes, and sampled approved fixtures over raw prompts or outputs.

Each signal names purpose, sensitivity, access, retention, aggregation, threshold, owner, action, and blind spot. Rare or high-consequence failures need protected review paths; unrestricted logging is prohibited. Privacy authority decides permissible collection. Monitoring distributions can drift and do not prove causality.

The tabletop failure begins after a hypothetical partial cohort: a tokenizer/runtime update lowers synthetic memory but changes truncation, degrading long Gujarati behavior. An adapter/template mismatch is a competing explanation. The first action is contain: stop ramp, route to the prior qualified fallback, preserve privacy-minimized identities and traces, and open an incident timeline. Do not retrain or retune before isolating the layer.

Diagnose by layer

Incident diagnosis must distinguish data, checkpoint or adapter, tokenizer or template, evaluator, runtime, and infrastructure changes. [CLM-050] Some failures cross layers, so test hypotheses rather than assigning blame.

The diagnostic ladder asks for the next discriminating evidence:

- input/data: distribution, language, authorization, freshness, malformed state;
- tokenizer/template: revision, serialization, truncation, special tokens, padding;
- retrieval/context: source snapshot, permissions, conflicts, assembly, missing evidence;
- base/adapter/merge: exact digests, binding, checkpoint, merge state;
- decoding: sampling, stops, schema constraints, fallback;
- runtime/kernel: version, precision, quantization, kernel path, cache, scheduler;
- evaluator: prompt/model/version, calibration, disagreement, slice coverage;
- product integration/infrastructure: routing, timeout, retries, state, capacity.

Replay the same frozen cases while changing one suspected surface. The tabletop compares old and new tokenizer serialization under the same package plan and finds earlier truncation of Gujarati citation evidence. Restoring the pinned tokenizer removes the deterministic failure. This supports a tokenizer/runtime cause in the fixture; it is not a production diagnosis.

V2-F17.2 - Change is complete only when evidence and controls replay. Essential labels: detect, contain, diagnose, correct, verify, replay, decide, rollback, deprecate, learn. Direction and state shapes supplement color. Supports CLM-050 and CLM-051.

Text alternative: A controlled loop moves through detect, contain, diagnose by layer, correct, verify, replay, authority decision, rollback or ramp, deprecate, and learn.

Correct, verify, and preserve rollback

A correction is incomplete until evidence, tests, controls, manifests, limitations, and monitoring are updated and replayed. [CLM-051] Pinning the prior tokenizer is containment or rollback, not necessarily the long-term correction. A forward fix creates a new complete identity and replays compatibility, target, retention, language, control, workload, degradation, and recovery evidence.

The rollback record binds the previous package, trigger, routing change, state reconciliation, verification cases, owners, and outcome state. Unknown external effects require reconciliation before retry. Mosaic's tabletop performs no effect and records rollback as simulated.

Deprecation needs last-supported identity, replacement or fallback, migration evidence, owner, archive/retention, and unresolved users. Provider deprecation schedules are volatile dependency inputs. A managed fallback is versioned and requalified under the same external behavior contract, without invented internals.

Keep authority distributed

LLM engineering owns model-system evidence, layered diagnosis coordination, and replay recommendations. Platform/SRE owns capacity, routing, reliability, and operational recovery. Incident command owns incident decisions and communications. Product/domain/language authorities own behavior consequence; privacy owns observation; security owns security response; legal owns legal interpretation; release authority issues ramp, hold, rollback, or deprecation.

The engineer cannot self-approve because tests pass, and a monitoring dashboard cannot accept residual risk. Disagreement remains in the record with evidence, owner, deadline, and escalation path.

Close MD-16 and the series

The deterministic release artifact links MD-01 through MD-16, issues a hold for the blocked package, runs a tabletop tokenizer/runtime incident, demonstrates layered diagnosis and rollback evidence, and updates tests, card, limitations, monitoring, and change ledger. No Mosaic outcome is claimed.

Portfolio learning separates durable reusable controls from local choices. Reusable candidates include manifest identity, compatibility gates, protected-slice replay, privacy-minimized signal contracts, and layered diagnosis. Local data, thresholds, language judgments, capacity, and release decisions do not become universal standards from one fictional case.

The closing case set remains bounded: `LLME-CASE-008` supplies staged-recipe lineage, `LLME-CASE-009` adapter compatibility, `LLME-CASE-011` serving workload reasoning, `LLME-CASE-012` compression requalification, `LLME-CASE-013` lifecycle change, and `LLME-CASE-014` untrusted-content risk. None supplies a Mosaic release, incident, safety, or capacity outcome.

Volume 1 taught how to define, build, evaluate, release, and change observable LLM behavior. Volume 2 began only after the MD-08 adaptation referral, ruled out upstream defects, built data and experiment evidence, rejected unjustified optimization, preserved package truth, and closed with a reversible hold. The final lesson is not “always tune.” It is to make every change earn its place in a reconstructable behavior system.

Practice

Run the tabletop from detection through hold or rollback. Pass when the causal layer is supported by controlled replay, privacy-minimized signals remain adequate, rollback is reconstructable, formal owners sign their decisions, uncertainty stays visible, and the entire MD-01 to MD-16 chain can be rebuilt.

Appendices

Adaptation measurement and optimization recall, model/data/artifact identity, reusable Mosaic dossier artifacts, provider-neutral companion guidance, inference and distributed-systems recall, terminology and adjacent-role boundaries, and publication provenance records.

Appendix A - Adaptation Measurement and Optimization Refresher

This appendix recalls the concepts used to interpret the volume's decisions. It is not a substitute for foundational study in probability, statistics, linear algebra, optimization, or deep learning. The examples remain synthetic and do not prescribe a real training configuration.

Begin with a behavior claim

An adaptation claim should identify:

Field	Question
contract	which required or prohibited behavior is being tested?
baseline	which exact model-system version is the comparator?
intervention	what single surface changed?
population	which intended cases and segments are represented?
measure	which deterministic check, rater, judge, or resource instrument produced the observation?
uncertainty	what variation, disagreement, or missing coverage remains?
protection	which retention and control cases may not regress?
disposition	retain, revise, reject, scope, hold, or send to release review?

Optimization loss, preference score, benchmark score, and product outcome are different claims. None should silently stand in for another.

Tokens, examples, and loss regions

A tokenizer maps text to token identifiers under a named revision. The template maps roles and fields to a serialized token sequence. Changing either can alter length, truncation, control tokens, padding, labels, and which positions contribute to the objective.

For supervised next-token training, the usual sequence loss averages negative log probability over selected target positions. A loss mask decides which positions count. The engineering record must preserve tokenizer, template, truncation, padding, special tokens, label construction, loss mask, and denominator. Two runs with different effective target regions are not clean comparators even if both report "training loss."

Perplexity is an exponential transformation of average token loss. It remains tokenizer-, corpus-, and implementation-dependent. It does not directly measure citation validity, abstention, schema compliance, product value, or safety.

Gradient and optimizer state

Gradient-based optimization estimates a direction that changes parameters to reduce an objective on sampled batches. Learning rate, schedule, optimizer, weight decay, clipping, precision, accumulation, batch order, and random seed all affect the trajectory.

Effective batch size should be stated in the implementation's actual units. A common data-parallel form is:

```
examples per device x devices x gradient accumulation steps
```

Sequence packing, variable lengths, token-based batching, dropped batches, and distributed reduction semantics can make that shorthand incomplete. Record the measured tokens, examples, updates, and world size rather than trusting one configuration label.

A resumable checkpoint may need weights or adapters, optimizer state, scheduler state, scaler state, random-number-generator state, data cursor, sampler state, and the exact code/config identity. Loading weights alone is not necessarily a faithful resume.

Full adaptation and parameter-efficient adaptation

Full fine-tuning updates all or a broad set of model parameters. Parameter-efficient methods hold the base fixed and learn a smaller change such as adapters or low-rank matrices. A common low-rank update represents a weight change as the product of two smaller matrices, scaled by a declared factor.

Fewer trainable parameters can reduce some training-memory and storage costs, but it does not prove better behavior, faster inference, merge equivalence, lower operational cost, or compatibility. Rank, scale, dropout, target modules, base revision, tokenizer, template, precision, quantization, merge state, and runtime all belong to the candidate identity.

Preference objectives

Preference methods learn from comparisons or scores. Their objective is a proxy for the desired behavior, not the behavior contract itself. Pair construction, reference policy, temperature or regularization, position randomization, rater population, criterion wording, disagreement, verbosity/style bias, and judge coupling can change the learned shortcut.

Evaluate a preference candidate with independent target, retention, language, control, and resource evidence. If the same judge defines the pairs and declares success, record the coupling rather than calling the result independent validation.

Distillation and continued pretraining

Distillation transfers behavior or distributions from a teacher into a student. It introduces teacher limitations, generation or logit identity, filtering, rights, cost, and refresh dependencies. Continued pretraining changes a model on additional unlabeled or weakly structured text. It may target domain representation or language modeling rather than the typed behavior contract.

Neither is "more advanced SFT." Each requires a distinct mechanism hypothesis, data and rights argument, resource envelope, protected evaluation, and rollback plan. Reject the method when the residual is better explained by an interface, retrieval, label, evaluation, or runtime defect.

Precision, quantization, and numerical comparison

Precision changes storage, arithmetic, memory, speed, numerical stability, and sometimes behavior. Quantization maps values to a lower-precision representation using a declared method and, where relevant, a calibration artifact. "Four bit" is not a complete configuration.

Compare variants on the same model-system tuple, inputs, workload, runtime conditions, and protected evaluation. Exact output equality may be inappropriate for stochastic generation; vague semantic similarity is also insufficient. Predeclare deterministic fields, tolerances, behavioral gates, repeated trials, and adjudication rules.

Target, retention, and control evidence

Report target gain beside retention and control behavior. Useful lanes include:

- the diagnosed residual and its language or domain segments;
- ordinary in-distribution behavior that should remain stable;
- rare but consequential cases;
- abstention, refusal, escalation, and evidence-conflict states;
- schema, citation, permission, privacy, and safety controls;
- satellite cases that challenge over-specialization;
- resource and compatibility consequences.

An aggregate improvement cannot compensate for a release-blocking protected regression unless the designated authority explicitly changes the contract. The engineer cannot make that trade by renaming the aggregate.

Repeated evidence and uncertainty

Random seeds, sampling, batch order, non-deterministic kernels, distributed execution, and judge variability can change observations. Repeat the method at the level relevant to the claim and retain the distribution, not only the best run.

A confidence interval or variability band does not repair a biased population, contaminated split, invalid judge, or changed configuration. Uncertainty language should name the source of uncertainty and the population to which the estimate applies.

Resource measurements

Training and inference resource claims require explicit units and boundaries. Record wall time, device time, peak allocated and reserved memory where applicable, tokens or examples processed, energy or cost method when claimed, storage, checkpoint count, network assumptions, warmup, concurrency, and workload distribution.

The synthetic resource units in the companion demonstrate comparison logic only. They are not GPU measurements, vendor prices, capacity forecasts, or service-level objectives.

Decision checklist

Before promoting an adaptation result, confirm:

1. the baseline and candidate identities are immutable and comparable;
2. the data population, split, rights, and rendering are reproducible;
3. the intervention and alternative explanations were preregistered;
4. loss and proxy metrics are separated from held-out behavior;
5. target, retention, language, control, compatibility, and resource gates all ran;
6. uncertainty, disagreement, missing coverage, and failed candidates remain visible;
7. the artifact exists and its lineage is verified;
8. the runtime envelope matches the intended workload;
9. rollback restores a complete prior model system;
10. formal owners, specialist reviews, and release authority remain separate.

Appendix B - Model, Data, and Artifact Identity

Adaptation evidence is interpretable only when every behavior-facing object has a stable identity. This appendix defines a minimum content-addressed record. It does not prescribe a registry product, storage system, or security control.

Model-system tuple

Bind these components as one candidate identity:

Component	Minimum identity
base weights	publisher, family, immutable revision, digest, format, dtype
tokenizer	immutable revision, digest, vocabulary and special-token behavior
template	digest, allowed roles, serialization, golden token sequence
adapter or delta	method, config, base binding, digest, merge state
schema	request/response version and digest
generation defaults	decoding and stop configuration digest
runtime	engine, version, build, device/driver compatibility
precision/quantization	method, parameters, calibration identity, accumulation policy
control configuration	validation, permission, abstention, escalation, routing identity

A friendly model name or mutable registry tag can point to this tuple, but it cannot replace it. Resolve mutable names to immutable revisions before an experiment and preserve the resolution evidence.

Integrity is not compatibility

A valid checksum establishes that bytes match the recorded bytes. It does not establish that:

- the tokenizer belongs to the weights;
- the template matches training and serving expectations;
- an adapter was trained against the base revision;
- the runtime supports the artifact format and operations;
- precision or quantization preserves protected behavior;
- the license or purpose permits the intended use;
- the package has release approval.

Run integrity, provenance, compatibility, behavior, security, rights, runtime, and authority gates separately. Preserve which gate stopped the transition.

Data identity

A dataset version should identify the recipe, source snapshot, rights/purpose state, transforms, code, accepted and rejected record manifests, deduplication method, family grouping, split assignment, rendering template, tokenizer, loss mask, and aggregate counts.

Store content identities for records or bounded shards where policy permits. A row count and filename are not sufficient. If raw content cannot be retained, preserve the permitted metadata, transformation evidence, deletion lineage, and limitations needed for audit.

Split identity and leakage barriers

Train, development, evaluation, retention, and control partitions serve different decisions. Their manifest should record:

- record and family identifiers;
- source and time boundaries;
- exact and near-duplicate results;
- cross-partition overlap checks;
- benchmark and prompt-template overlap checks where observable;
- inaccessible or unknowable pretraining overlap;
- frozen versus refreshable status;
- owner and change trigger.

Passing a known-overlap check proves only that the implemented method found no prohibited overlap in the inspected artifacts. It does not prove global non-contamination.

Run identity

A training run manifest should bind:

- code commit or immutable source bundle;
- environment and dependency lock;
- model-system tuple;
- data and split manifests;
- objective, target construction, and loss mask;

- optimizer, schedule, precision, clipping, regularization, and seeds;
- batch, accumulation, packing, sequence, and distributed configuration;
- logging, checkpoint, evaluation, stop, and recovery policy;
- device and resource boundary;
- authority, limitations, and current disposition.

Every checkpoint points back to the run and forward to its evaluations. A resume creates a lineage edge that names the checkpoint and restored state. A failed resume remains in the record.

Checkpoint and adapter lineage

For each checkpoint, adapter, merge, export, or quantized variant, preserve:

Field	Purpose
parent	exact upstream artifact or artifacts
operation	training step, merge, conversion, quantization, or export
configuration	parameters and tool/runtime versions
digest	bytes and digest algorithm
compatibility	model/tokenizer/template/runtime results
evaluation	target, retention, control, and resource evidence links
limitations	missing tests, unsupported hardware, unresolved rights, or unknown behavior
state	candidate, rejected, blocked, review, promoted, deprecated, or revoked

Merged and unmerged adapters are distinct artifacts. Exported and quantized forms are distinct artifacts. Do not assign a pretend digest to an artifact that was not created.

Package manifest

A release candidate package should inventory every required component and fail closed when one is absent. It should include lineage, checksums, source and model cards, licenses and notices, compatibility matrix, protected evaluation, workload/resource evidence, security and privacy review references, known limitations, promotion state, rollback identity, and named owners.

The package status is not approval. `complete`, `compatible`, and `behavior-qualified` are evidence states; `release-approved` requires the designated authority.

Rollback identity

Rollback restores the prior complete model system, not just earlier weights. Record prior weights, tokenizer, template, adapter, schema, defaults, runtime, control configuration, routing, and data/evaluator versions used for verification. Name state that cannot be rolled back and the required reconciliation path.

Change replay

For checkpoint, tokenizer, template, runtime, quantization, or router change:

1. freeze the current and candidate tuples;
2. state the reason and expected behavior/resource effect;
3. resolve licenses, provenance, rights, and security gates;
4. replay compatibility and golden serialization;
5. replay target, retention, language, control, and failure cases;
6. run workload-specific resource qualification;
7. compare shadow or bounded-cohort evidence if authorized;
8. preserve stop, fallback, rollback, and reconciliation;
9. record disposition and next trigger.

If a component cannot be identified or replayed, the package is not change-ready.

Appendix C - Adaptation and Runtime Artifacts and Review Gates

These templates summarize the Mosaic dossier from `MD-09` through `MD-16`. They are starting points for review, not universal forms, controls, approvals, or compliance evidence.

MD-09 - Adaptation gate

Record:

- inherited behavior contract, baseline, evaluation, budgets, release state, and unresolved defects;
- new residual population, repeated evidence, first divergent layer, and alternative explanations;
- falsifiable adaptation hypothesis, expected target, protected behavior, rejection rules, and stop conditions;
- immutable model/tokenizer/template/runtime candidate tuple;
- compatibility results and blockers;
- intervention ladder from no-change and system repair through weight change and replacement;
- access, reversibility, resources, rights, specialists, owners, and authority;
- disposition: reject, investigate, data-plan-only, or training review.

Gate: training cannot begin while the residual is not stable, smaller interventions are untested, the artifact tuple is incompatible, or required authority is absent.

MD-10 - Data recipe and separation

Record:

- example unit, population, source, purpose, rights, privacy class, retention, deletion, and owner;
- target and protected segment balance;
- inclusion, exclusion, quality, normalization, and rejection rules;
- transforms with code/config identities;
- exact and near-duplicate families;
- train, development, evaluation, retention, and control split manifests;
- overlap checks, inaccessible overlap, contamination limitations, and freeze/refresh policy;
- accepted and rejected counts with reasons;
- disposition and unresolved data authority.

Gate: a recipe can pass mechanically while the data remain insufficient, unrepresentative, unauthorized, or blocked.

MD-11 - Instruction, preference, and protected evidence

Instruction record:

- behavior clause and source evidence;
- input, authorized context, expected output or terminal state;
- tokenizer/template render identity;
- target and loss-mask regions;
- language, domain, consequence, evidence state, and difficulty tags;
- reviewer, checks, limitations, and split assignment.

Preference record:

- admissible candidates and criterion-specific comparison;
- randomized display order and length/style controls;
- rater qualification, independence, disagreement, and adjudication;
- source evidence and privacy handling;
- known shortcut and coupling risks.

Protected suite:

- target, retention, language, abstention, escalation, safety, permission, citation, schema, and satellite-case lanes;
- deterministic, human, domain, model-judge, or authority method for each claim;
- hard gate versus diagnostic measure;
- owner and change trigger.

Gate: learning data do not become training-ready while rendering, rights, coverage, or protected evidence remain blocked.

MD-12 - Training and candidate comparison

Run card:

- question, baseline, intervention, and alternative explanations;
- exact model-system and dataset identities;
- objective, optimizer, schedule, precision, seeds, batch/packing, checkpoints, logging, resources, and recovery;

- smoke, stop, evaluation, and promotion gates;
- executed versus planned state.

Candidate matrix:

Lane	No tune	SFT	PEFT	Required disposition
diagnosed target	observation	observation	observation	minimum gain or reject
retention	observation	observation	observation	no prohibited regression
language segments	observation	observation	observation	segment gates preserved
controls	observation	observation	observation	all hard gates pass
compatibility	observation	observation	observation	exact tuple passes
resources	observation	observation	observation	within declared envelope

Gate: loss improvement, trainable-parameter count, or one aggregate score cannot promote a candidate.

MD-13 - Preference and advanced-method decision

Record:

- residuals after the smallest credible candidate;
- whether preference evidence adds a distinct criterion;
- data bias, judge/rater coupling, reference behavior, and protected evaluation;
- distillation teacher/student thesis, transfer artifact, rights, costs, and failure modes;
- continued-pretraining corpus thesis, tokenizer/domain evidence, contamination, rights, compute, and forgetting risks;
- system repair or replacement alternatives;
- execution state, rejected methods, limitations, and next trigger.

Gate: method sophistication is not evidence. Reject any method whose mechanism does not match a distinct residual.

MD-14 - Model-system package

Inventory:

- weights, tokenizer, template, adapter, schema, defaults, runtime, precision/quantization, controls, and routing;
- content identities, lineage, licenses, notices, and provenance;
- required versus present state;
- integrity and compatibility results;

- target, retention, control, and resource evidence;
- security, privacy, platform, product/domain, and release review references;
- limitations, promotion state, fallback, and complete rollback tuple.

Gate: missing artifacts remain missing. A valid hash with an incompatible tuple blocks before load.

MD-15 - Inference and serving envelope

Workload card:

- input and output token distributions;
- concurrency, arrival pattern, burst and queue assumptions;
- prefill and decode timing;
- weight, KV, activation, allocator, and overhead boundaries;
- batch, cache, routing, parallelism, and loading configuration;
- latency percentiles, throughput, error, saturation, and degradation behavior;
- target, retention, language, and control replay for every serving variant;
- hardware, runtime, measurement window, warmup, and exclusions.

Frontier record:

- candidate variants and exact tuple identities;
- behavior and resource observations with uncertainty;
- dominated and rejected candidates;
- fallback ladder and expiry trigger;
- specialist owner and capacity limitation.

Gate: a fast aggregate cannot erase a protected regression or unsupported tail/capacity claim.

MD-16 - Release, incident, and change record

Readiness packet:

- complete dossier index and digests;
- current package and serving tuple;
- passed, failed, missing, and expired gates;
- intended cohort, exclusions, exposure, signals, stop triggers, fallback, rollback, owner, and authority;
- explicit go, conditional go, reduce scope, delay, or hold disposition.

Incident/change record:

- detected symptom, affected version and segment, first safe action, and authority;
- privacy-minimized timeline of confirmed facts, unknowns, hypotheses, and evidence requests;
- layered diagnosis across task, data, retrieval, prompt/template, tokenizer/model, validation, evaluator, runtime, capacity, product, and observation;
- containment, rollback or roll-forward, reconciliation, and recovery evidence;
- checkpoint/tokenizer/runtime migration replay;
- durable changes to evaluation, controls, card, monitoring, ownership, and limitations.

Gate: a reconstructed dossier can correctly end in hold. No artifact or engineer approves itself.

Review order

Use this order so downstream work cannot hide an upstream gap:

1. behavior and authority contract;
2. residual and causal layer;
3. artifact compatibility;
4. data purpose, rights, population, and separation;
5. experiment identity and execution state;
6. target, retention, language, control, and uncertainty;
7. package integrity and compatibility;
8. workload-specific resource qualification;
9. security, privacy, platform, product/domain, and release gates;
10. exposure, signals, stop, recovery, and change replay.

Appendix D - Provider-Neutral Companion Guide

What the companion proves

The Volume 2 companion is a deterministic local implementation of selected Mosaic Desk adaptation and runtime records. It proves that its JavaScript functions and supplied synthetic fixtures behave as asserted by the included tests.

It does not prove:

- that any model was executed or trained;
- that any checkpoint, adapter, quantized model, or deployable package exists;
- that the synthetic data are representative, authorized for a real use, or sufficient;
- that an optimization method improves real behavior;
- that a model, tokenizer, template, adapter, or runtime is compatible outside the fixtures;
- that resource units correspond to hardware, latency, throughput, capacity, energy, or cost;
- that a release, incident, rollback, or migration occurred;
- that product, domain, privacy, security, legal, platform, risk, or release authorities approved anything.

Fields such as `trainingExecuted`, `modelExecuted`, `benchmarkExecuted`, and artifact-creation state are evidence-bearing. Do not remove or invert them to make a demonstration look more complete.

Commands

From the repository root:

```
node --test content/publications/llm-adaptation-and-runtime/companion/tests/*.test.mjs
node content/publications/llm-adaptation-and-runtime/companion/run.mjs
```

The companion uses Node.js built-ins, local JSON, deterministic code, and no provider credential, model download, network call, training framework, or accelerator.

Artifact map

Dossier	Directory	Purpose
MD-09	`companion/md09/`	frozen baseline, model/tokenizer inspection, adaptation ladder

Dossier	Directory	Purpose
MD-10	`companion/md10/`	data recipe, curation, duplicate/overlap, split manifests
MD-11	`companion/md11/`	instruction, preference, retention, and control evidence
MD-12	`companion/md12/`	blocked training plan and simulated SFT/PEFT comparisons
MD-13	`companion/md13/`	rejected preference and advanced-method simulations
MD-14	`companion/md14/`	incomplete fail-closed model-system package
MD-15	`companion/md15/`	capacity and serving-frontier simulations
MD-16	`companion/md16/`	release hold, tabletop diagnosis, rollback, and change replay

The library modules implement the mechanics. The chapter tests define the local failure oracle. The JSON records are committed evidence outputs, not hand-edited success summaries.

Safe extension rules

An extension should:

1. begin from a named behavior clause and current dossier version;
2. use synthetic or explicitly authorized data only;
3. bind model, tokenizer, template, adapter, runtime, and configuration identities;
4. preserve rejected records, failed candidates, gaps, and unknowns;
5. keep target, retention, language, and control evidence separate;
6. distinguish planned, simulated, executed, and created states;
7. avoid raw sensitive content in logs and artifacts;
8. fail closed on missing identity, authorization, compatibility, or authority;
9. add a failing test before changing a transition;
10. state what the result cannot establish.

Do not add live provider or training calls merely for realism. If an authorized experiment is added in another environment, keep its credentials, restricted data, model artifacts, infrastructure, and raw traces outside this publication repository. Import only the permitted, content-addressed evidence summary and limitations.

Adding an adaptation method

Define the mechanism hypothesis and distinct residual first. Add its data requirements, artifact tuple, objective, configuration, execution state, resource boundary, target and protected evaluation, alternative explanations, stop rules, rollback consequence, and rejection result. Compare it against no-change and the smallest credible intervention.

Adding a dataset transform

Give the transform an immutable code/config identity. Preserve source, purpose, rights, record family, accepted/rejected state, reason, prior and resulting digest where permitted, split boundary, and deletion lineage. Re-run overlap, rendering, coverage, retention, and control checks.

Adding a runtime variant

Bind the exact model-system tuple and workload. Record warmup, request/input/output distribution, concurrency, queue assumptions, memory boundary, percentiles, throughput, errors, saturation, and protected behavior replay. A synthetic calculation should remain labeled synthetic; a local benchmark should remain bounded to its hardware and method.

Adding an evaluator

State the criterion, population, reference or rubric, judge identity, prompt/configuration, calibration, disagreement, bias risks, coupling, cost, privacy, and what the evaluator cannot establish. Preserve raw content only when authorized. Do not promote model-judge agreement into domain truth or formal approval.

Reproducibility and proof records

When companion output changes, record the source identity, command, environment boundary, tests, output digest, prior digest, reason, reviewer, and limitations. A deterministic rerun supports local reproducibility only. It does not establish external validity or production readiness.

Appendix E - Inference and Distributed-Systems Refresher

This appendix supports Chapters 14-17 at decision depth. It does not replace platform, performance, accelerator, networking, database, reliability, security, or SRE expertise.

The model-system request path

An inference request can cross admission, routing, tokenization, model loading, scheduling, prefill, decode, validation, retrieval or tools, caching, human review, persistence, and observation. End-to-end behavior depends on each version and timeout boundary, not only the model kernel.

Draw the path with:

- principal, tenant, region, and authorization;
- request and response schema versions;
- tokenizer, template, model, adapter, precision, runtime, and router identities;
- queue and timeout boundaries;
- mutable state and caches;
- retries and idempotency;
- validators, fallback, and human gates;
- privacy-minimized signals;
- owner and recovery action at each layer.

Weight, activation, and KV memory

Weight memory depends on parameter count, representation, quantization metadata, replicas, sharding, and runtime overhead. Active inference also needs workspace, allocator, graph, kernel, activation, communication, and cache memory.

Autoregressive attention commonly retains key/value state for prior tokens. KV demand grows with active sequences, retained context, layers, hidden/head dimensions, representation, and implementation. Prefix sharing, paging, eviction, offload, and quantization can change the boundary, but each introduces compatibility, latency, fragmentation, or behavior questions.

Do not infer safe concurrency from weight fit alone. Measure the complete runtime under the intended context and output distributions.

Prefill and decode

Prefill processes the input context and often uses parallel matrix operations. Decode produces tokens sequentially per sequence while schedulers batch work across requests. Their latency and resource characteristics differ.

Report end-to-end latency as well as time to first token and inter-token or generation timing when relevant. A short prompt/long output workload and a long prompt/short output workload can have similar token totals and very different behavior.

Batching and scheduling

Static batching waits for a group and can add queue delay. Dynamic or continuous batching admits work as sequences arrive or complete. Larger batches may improve device utilization while increasing latency, memory pressure, interference, and tail risk.

Record admission, queue discipline, priorities, maximum tokens, cancellation, fairness, padding or packing, preemption, and overload behavior. Throughput at saturation is not the same as capacity under a latency objective.

Caching

Result caching, prefix caching, and KV reuse solve different problems. A cache key must bind every behavior-bearing input and authorization boundary. Include model, tokenizer, template, adapter, schema, decoding, context/evidence version, principal or permission scope, and relevant control configuration.

Caching can leak data, serve stale evidence, cross tenants, hide a regression, or make benchmark traffic unrepresentative. Define freshness, invalidation, isolation, eviction, observation, and fail-closed behavior. A cache hit is not evidence that the underlying model would still pass.

Routing and multiple variants

A router may choose among models, adapters, precisions, replicas, regions, or fallbacks. The routing rule becomes part of the behavior contract. Record eligibility, feature inputs, confidence or threshold semantics, excluded segments, health signals, capacity signals, audit fields, and fallback behavior.

A cheaper or faster variant must pass its own target, retention, language, control, compatibility, and workload gates. Aggregate router quality can hide a failing segment.

Parallelism and specialist boundaries

Data, tensor, pipeline, sequence, or expert parallelism can change memory, communication, failure recovery, numerical behavior, and operational complexity. The LLM Engineer should understand their task-level consequences and evidence needs. Infrastructure and performance specialists own topology, kernel, collective, scheduler, and hardware qualification.

Escalate when conclusions depend on device topology, interconnect saturation, kernel selection, compiler behavior, allocator fragmentation, multi-node recovery, or production capacity. Do not convert a conceptual calculation into a platform promise.

Queues, overload, and tail behavior

Arrival bursts, long contexts, slow decodes, retries, and unavailable replicas can amplify queue length. Average latency can remain acceptable while a consequential segment encounters severe tails.

Define admission and load-shedding policy, maximum queue time, request budget, cancellation, retry ownership, degradation ladder, backpressure, and recovery. A retry consumes capacity and can worsen the incident. Unknown completion requires reconciliation before repeating any external effect.

Capacity evidence

A capacity claim should name:

- workload population and time window;
- input/output/context distributions and segments;
- concurrency, arrival, burst, and cache assumptions;
- exact package, runtime, hardware, topology, and region;
- warmup, load duration, percentiles, errors, saturation, and exclusions;
- memory boundary and observed headroom;
- behavior replay under load;
- failure and recovery behavior;
- owner, expiry, and next trigger.

Synthetic examples can validate equations and state transitions. Local tests can qualify one environment. Neither automatically predicts production traffic or hardware.

Timeouts, retries, and unknown outcomes

Timeouts bound waiting, not necessarily work. A client can time out while the server continues. Retrying a non-idempotent operation can duplicate an effect. Use an idempotency key tied to stable intent, reconcile unknown outcomes, and separate model proposal from authorized action.

Mosaic Desk stops before external effect. Its runtime records still model timeout, fallback, and reconciliation so an integrating product team can preserve that boundary.

Observation without raw-content default

Useful signals often include version identifiers, state transitions, segment IDs, reason codes, latency buckets, token buckets, resource buckets, validator outcomes, fallback path, and correlation identifiers. Raw prompts, evidence, outputs, labels, gradients, or traces may contain sensitive or restricted data.

Collect only what has a named purpose, owner, retention period, access policy, decision, blind spot, and deletion behavior. Privacy-minimized telemetry is not automatically sufficient or approved; it is a design input for the responsible specialists.

Layered diagnosis

Investigate regressions across:

1. task and population change;
2. data recipe, split, rendering, or rights change;
3. retrieval or authorized evidence change;
4. prompt, template, tokenizer, model, adapter, or decoding change;
5. schema, validation, evaluator, or human-review change;
6. precision, quantization, runtime, kernel, router, cache, or capacity change;
7. product workflow, permissions, or observation change.

Ask for the next discriminating evidence. Do not optimize the first plausible layer before consequence is contained.

Release and rollback

Release uses immutable artifacts, bounded exposure, shadow or canary evidence where authorized, named stop triggers, privacy-minimized observation, complete fallback, and formal authority. Rollback must restore the whole prior tuple and reconcile state that cannot be reversed.

After recovery, replay the contract, update the incident and change ledger, expire invalid evidence, and record what remains unknown. Restoration alone is not durable learning.

Appendix F - Terminology and Adjacent-Role Boundaries

Adaptation and data terms

- **adaptation:** a deliberate change intended to alter model-system behavior; it may or may not change weights.
- **fine-tuning:** continued optimization of some or all model parameters on a specified objective and dataset.
- **supervised fine-tuning (SFT):** token-level optimization against constructed target sequences.
- **parameter-efficient fine-tuning (PEFT):** adaptation that learns a smaller parameter delta or module while retaining a base model.
- **adapter:** a learned component applied to a particular base and compatibility tuple.
- **preference optimization:** optimization using comparative or scored behavior evidence as a proxy objective.
- **distillation:** transfer from a teacher signal into a student under a declared objective.
- **continued pretraining:** additional language-model training on a specified corpus, distinct from direct instruction targets.
- **data recipe:** versioned definition of population, sources, rights, purpose, transforms, quality, exclusions, splits, retention, and outputs.
- **example family:** records related closely enough that cross-split placement could create leakage.
- **contamination:** prohibited or decision-invalidating overlap between learning evidence and evaluation, benchmark, or protected evidence.
- **retention case:** evidence for behavior the adaptation should preserve.
- **control case:** evidence for required fail-closed, abstention, escalation, permission, safety, privacy, schema, or authority behavior.
- **synthetic data:** constructed data whose generation, review, representativeness, and rights limitations remain explicit.

Model and experiment terms

- **base model:** immutable weights identity against which an adapter or adapted checkpoint is defined.
- **checkpoint:** captured training state at a named step; it may include more than weights.
- **model-system tuple:** weights, tokenizer, template, adapter, schema, defaults, runtime, precision, controls, and routing that jointly affect behavior.

- **template:** deterministic serialization from structured messages or examples into model-facing tokens or text.
- **loss mask:** rule selecting target positions that contribute to an optimization objective.
- **effective batch:** actual examples or tokens contributing to an update across packing, accumulation, devices, and distributed semantics.
- **ablation:** controlled comparison removing or changing one component to test its contribution.
- **proxy metric:** a measure related to the target decision but not identical to the behavior or product claim.
- **hard gate:** a criterion whose failure blocks the declared transition.
- **simulated candidate:** deterministic teaching data used to exercise comparison logic without model execution or artifact creation.
- **lineage:** directed record of parent artifacts, operations, configurations, and resulting identities.
- **merge:** operation applying or combining an adapter with a base; merged and unmerged forms are distinct artifacts.

Runtime terms

- **prefill:** processing of the input context before autoregressive generation.
- **decode:** iterative generation of output tokens.
- **KV cache:** retained key/value attention state used across decode steps.
- **continuous batching:** scheduling that admits and removes sequences dynamically during generation.
- **prefix cache:** reusable state for an identical authorized prefix under a complete behavior-bearing key.
- **quantization:** representation and arithmetic change using lower precision under a named method and configuration.
- **throughput:** completed work per time under a stated workload and boundary.
- **latency tail:** high-percentile delay that may reveal queue, context, segment, or overload behavior hidden by an average.
- **capacity:** workload that a complete system can sustain under declared behavior, latency, error, resource, and recovery objectives.
- **behavior-resource frontier:** non-dominated candidates considered jointly on protected behavior and resource evidence.
- **graceful degradation:** predefined reduction in service that preserves critical behavior and authority rather than failing unpredictably.

Evidence and lifecycle terms

- **integrity:** evidence that bytes match a recorded digest.

- **compatibility:** evidence that components can be combined under a named interface and runtime.
- **qualification:** bounded evidence that a complete candidate meets stated gates for a stated population and environment.
- **promotion:** state transition in an artifact or deployment lifecycle; it is not automatically release approval.
- **shadow:** execution whose outputs are withheld from the user or external effect while evidence is collected under authorization.
- **canary or bounded cohort:** limited exposure with named eligibility, signals, stop triggers, fallback, and authority.
- **rollback:** restoration of a prior complete model-system tuple with reconciliation of non-reversible state.
- **requalification:** replay required after a checkpoint, tokenizer, template, runtime, quantization, data, evaluator, or workload change.
- **hold:** explicit disposition that prevents the next transition while preserving the evidence and trigger for reconsideration.

Adjacent-role boundaries

LLM Engineer

Owns the language-behavior contract, model-system evidence, adaptation hypothesis, target and protected evaluation, and technical recommendation within delegated scope. Does not self-authorize data use, infrastructure risk, product value, or release.

Applied AI Engineer

Owns the broader AI-enabled product behavior and integration across model and non-model components. The role may consume adaptation evidence but does not make a training result sufficient for product readiness.

Machine Learning Engineer

May own training pipelines, feature/data systems, optimization implementation, artifact production, and ML lifecycle operations. The LLM Engineer supplies and evaluates the language-behavior contract; ownership is explicit per team.

AI Researcher

Develops or studies new objectives, architectures, algorithms, and empirical explanations. Engineering use of a published method does not transfer the paper's claims to Mosaic or turn the engineer into the research authority.

AI Evaluation Engineer

Designs and validates evaluation systems, judges, rater programs, populations, and measurement infrastructure. The LLM Engineer remains responsible for connecting those measures to the stated behavior decision.

Data owner or steward

Decides permitted source use, purpose, quality, retention, deletion, and access within organizational policy. Technical ability to copy data is not authorization to adapt a model with it.

Privacy and legal specialists

Interpret privacy obligations, rights, licenses, terms, retention, deletion, and jurisdiction. Dataset availability or an open-weight label is not sufficient legal approval.

Security and supply-chain specialists

Own threat analysis, artifact acquisition controls, scanning, signing policy, isolation, secret handling, and risk acceptance. Checksums support integrity but do not replace this work.

Platform, MLOps, and infrastructure engineers

Own registries, deployment systems, accelerators, schedulers, networking, capacity, availability, recovery, and supported runtime configurations. A task-level resource model is an interface to their work, not a production capacity claim.

Performance engineer

Owns rigorous workload design, profiling, kernel/runtime investigation, bottleneck analysis, and performance qualification. The LLM Engineer supplies behavior gates so acceleration does not silently change the product contract.

Reliability or SRE specialist

Owns service objectives, observability, incident systems, on-call readiness, capacity and recovery practices within the organization. The book's tabletop artifacts are not evidence that those systems exist.

Product and domain owners

Decide whether the target matters, what tradeoffs are acceptable, which behavior needs qualified human review, and whether a candidate serves the workflow. Model scores do not confer domain authority.

Safety, governance, and risk authorities

Define required controls, evidence, exceptions, and acceptance within their remit. The LLM Engineer implements and tests assigned controls but cannot accept residual risk for them.

Release authority

Makes the formal exposure decision using the complete readiness packet. The engineer can recommend go, conditional go, reduce scope, delay, or hold; the artifact cannot approve itself.

Boundary questions

Before acting, ask:

1. Is this a behavior-evidence decision or a formal organizational decision?
2. Which artifact and method support the claim?
3. Which specialist owns the unresolved technical boundary?
4. Who owns data purpose, rights, retention, and deletion?
5. Who decides product/domain value and acceptable tradeoffs?
6. Who accepts security, privacy, safety, infrastructure, or operational risk?
7. Who may authorize training, artifact handling, cohort exposure, or release?
8. What must stop if that authority or evidence is absent?

Appendix G - Source, Figure, and Edition Records

Publication identity

The canonical publication manifest is `publication.json`. It records the series and volume, title, author, language, publication state, edition version, chapter order, part boundaries, appendix summary, registry paths, canonical URL, and PDF configuration.

For this published edition:

- series: LLM Engineering;
- volume: 2;
- edition: 1.0.0, first edition;
- chapter count: 17;
- part count: 5;
- appendix count: 7;
- publication state: published;
- published date: 2026-08-17;
- canonical PDF: enabled for `llm-adaptation-and-runtime-v1.0.0.pdf`.

The manifest records the accepted publication decision. It does not approve a model, training job, package, deployment, production outcome, or organizational risk decision.

Source register

The canonical source register is `sources.json`. Each source has a stable identifier, title, organization or authorship, URL or locator, publication date where available, access date, source type, and bounded usage note.

The register supports traceability, not truth by association. A paper, standard, official documentation page, model card, dataset card, or vendor technical page can support a narrow mechanism, interface, or reported result. It does not establish Mosaic's population, data rights, hyperparameters, model compatibility, infrastructure, resources, or release readiness.

Source currentness reviews live under `production/`. They preserve the checked date, replacement risk, and disposition. A current URL does not make an old result current for a changed artifact or workload.

Claim register

The canonical claim register is `claims.json`. Stable claim IDs connect chapter statements to sources and retain scope notes. A claim record should make it possible to ask:

- what exact statement is supported;
- which source and section carry the evidence;
- whether the statement reports, explains, compares, or synthesizes;
- which model, data, runtime, version, task, and limitations apply;
- whether the chapter extends the source through an explicit inference.

Unsupported operational or population claims must not be smuggled into a sourced mechanism explanation.

Figure register and provenance

The canonical figure register is `figures.json`. The Volume 2 sequence is V2-F01.1 through V2-F17.2, with two figures per chapter. Each accepted record should preserve:

- stable figure ID and chapter anchor;
- title, caption, alt text, and long description;
- canonical PNG path and public mirror path;
- width, height, format, and SHA-256 identity;
- generation source and prompt/provenance record;
- required and forbidden labels;
- synthetic-art and evidence limitation;
- visual, semantic, typography, accessibility, and mirror QA disposition;
- correction or rejection history where applicable.

Figures are explanatory concept art. Their synthetic values, relative sizes, gauges, curves, frontiers, and states do not report measured model, training, product, user, resource, cost, or production outcomes. Color is supplemented by labels, geometry, gates, arrows, texture, icons, or position. The canonical source is PNG; delivery derivatives are not provenance sources.

Companion evidence

The companion records under `companion/md09/` through `companion/md16/` are deterministic teaching artifacts. Their source code, tests, JSON output, commands, and execution limits belong to the evidence record.

Passing tests establish local mechanics only. Simulated candidates remain simulated. `trainingExecuted: false`, `modelExecuted: false`, `benchmarkExecuted: false`, missing adapter state, template incompatibility, release hold, and other negative evidence must remain visible in summaries and proofs.

Errata

The canonical errata register is `errata.json`. An erratum should identify edition, location, problem, status, resolution, replacement text when applicable, and the edition or web update that carries the correction.

Review-stage corrections can be incorporated before publication while retaining review history. After publication, content must not change silently under the same edition identity. Material source, figure, code, or claim corrections require registry and proof review.

Deterministic PDF release

A release PDF should be generated from the same chapter order, front matter, five part introductions, seven appendices, figures, source register, and edition metadata used by the web edition. Record:

- source SHA-256 over the assembled publication inputs;
- PDF SHA-256 and byte count;
- page size and page count;
- chapter, appendix, figure, source, and errata counts;
- deterministic metadata and generated-time policy;
- bookmark and link inspection;
- text extraction and empty-page checks;
- full rendered-page inspection for clipping, overlap, missing glyphs, missing figures, and broken transitions;
- independent second build and byte comparison.

The final manifest must retain `pdf.enabled: true`, `status: published`, and `edition.publishedAt: 2026-08-17`. The installed canonical download must be byte-identical to the accepted release artifact. Any later material source or edition change requires a new reviewed artifact identity rather than silent replacement.

Complete-edition release

A truthful complete edition includes:

1. publication metadata and canonical order;
2. copyright, disclaimer, preface, audience, organization, evidence language, and companion limits;
3. all five part introductions and 17 chapters;
4. all seven appendices;
5. accepted figures with accessible descriptions and provenance;
6. source, claim, figure, and errata registries;
7. deterministic companion evidence and explicit execution limits;
8. web build and local-reference validation;
9. two deterministic final PDF builds with structural and visual inspection;
10. an explicit publication record and immutable download identity.

Publication does not imply production readiness, model approval, deployment approval, or organizational authority. It means this educational edition passed its declared source, manuscript, figure, companion, web, PDF, and release checks.

Release record

The deliberate release change for version 1.0.0:

- changed status from review to published;
- recorded 2026-08-17 as the publication date;
- enabled the canonical PDF;
- attached the approved PDF identity and release evidence;
- exposed the edition in published catalog and route surfaces.

Any source, figure, code, registry, metadata, or builder change after proof invalidates the prior source digest and requires the affected gates to rerun.

Figure registry

The 34 figures are original synthetic ImageGen PNG teaching illustrations for the fictional Mosaic Desk case. They explain adaptation, data, training, packaging, inference, serving, and change relationships; they do not report trained-model, product, user, capacity, or production outcomes.

FIG-001 - V2-F01.1 - Freeze the comparison unit before adaptation. Essential labels: frozen, candidate, contract, config, eval, budgets, controls. Shape and lock symbols supplement color. The figure supports `CLM-001`; it does not claim the baseline is production-approved.

Long description: A frozen baseline vault separates an identified baseline from a candidate and locks contract, configuration, evaluation, budgets, and controls before comparison.

FIG-002 - V2-F01.2 - A useful hypothesis can lose. Essential labels: error, mechanism, intervention, predicted evidence, rejection. Broken-link and stop-gate shapes supplement color. The figure supports `CLM-002` and `CLM-003`; it predicts no adaptation outcome.

Long description: A falsifiable chain connects an observed error to a proposed mechanism, intervention, predicted evidence, and explicit rejection gates.

FIG-003 - V2-F02.1 - Inspect separate surfaces inside one model bundle. Essential labels: tokens, embed, blocks, head, precision. Texture and component shape supplement color. The figure supports `CLM-004`; it is not a performance ranking.

Long description: A cutaway model system shows tokens entering embeddings and transformer blocks before a prediction head, with precision recorded as a separate runtime surface.

FIG-004 - V2-F02.2 - Token boundaries change the experiment surface. Essential labels: tokenizer A, tokenizer B, length. Brackets and count blocks supplement color. The figure supports `CLM-005`; it makes no multilingual quality claim.

Long description: The same Gujarati-English shorthand string is divided differently by tokenizer A and tokenizer B, producing visibly different sequence lengths without asserting which is better.

FIG-005 - V2-F03.1 - Climb only when the failure mechanism demands it. Essential labels: repair, SFT, PEFT, preference, distill, continue, replace. Stop signs and distinct platform shapes supplement color. The figure supports `CLM-007` and `CLM-009`; height does not mean quality or maturity.

Long description: A gated intervention ladder moves from repair through supervised and parameter-efficient tuning, preference optimization, distillation, continued pretraining, and replacement, with a stop gate at every rung.

FIG-006 - V2-F03.2 - Feasibility is a set of gates, not a price tag. Essential labels: access, data, compute, control, reversibility. Icons and narrowing rails supplement color. The figure supports `CLM-008`; it reports no actual budget or approval.

Long description: Intervention candidates are filtered through access, data, compute, control, and reversibility gates before one bounded pilot investigation remains.

FIG-007 - V2-F04.1 - A recipe turns sources into traceable examples. Essential labels: source, recipe, example, provenance. Shape and tab patterns supplement color. The figure supports `CLM-010`; documentation does not grant data rights.

Long description: Authorized source trays pass through a versioned recipe workbench into example records that retain source and transformation provenance tabs.

FIG-008 - V2-F04.2 - Planned balance makes convenience gaps visible. Essential labels: language, domain, error, abstention, gap. Patterns and empty tray outlines supplement color. The figure supports `CLM-011` and `CLM-012`; it reports no real population frequencies.

Long description: A target-population outline is compared with separate language, domain, error, and abstention trays, leaving convenience-sampling gaps visibly empty.

FIG-009 - V2-F05.1 - Indirect duplicates threaten split claims. Essential labels: train, dev, eval, overlap, false alarm. Line patterns and alarm shapes supplement color. The figure supports `CLM-013`; it does not prescribe a universal similarity threshold.

Long description: Near-identical record families approach train, development, and evaluation barriers; shared signatures trigger a false alarm while a paraphrased evaluation relative triggers a real overlap alarm.

FIG-010 - V2-F05.2 - Purpose-separated vaults preserve claim validity. Essential labels: train, dev, eval, retention, control. Distinct vault shapes and hash seals supplement color. The figure supports `CLM-014` and `CLM-015`; separation does not prove absence of unknown historical overlap.

Long description: Five physically separated hashed vaults hold train, development, evaluation, retention, and control records, with family links forbidden from crossing vault walls.

FIG-011 - V2-F06.1 - Show exactly what is supplied and learned. Essential labels: instruction, input, context, target, loss, metadata. Border and mask patterns supplement color. The figure supports `CLM-016` and `CLM-017`; it reports no training effect.

Long description: One instruction example separates trusted instruction, untrusted input, authorized context, typed target, intentional loss region, and provenance metadata.

FIG-012 - V2-F06.2 - Varied learning signals still leave honest holes. Essential labels: clause, error, positive, negative, abstain, multilingual, gap. Patch textures supplement color. The figure supports `CLM-016` and `CLM-018`; coverage is not a quality result.

Long description: A coverage quilt maps behavior clauses and error categories to positive, negative, abstention, and multilingual example families while leaving missing cells visible.

FIG-013 - V2-F07.1 - A preference is evidence about one rubric. Essential labels: A, B, rubric, disagree. Shape and order markers supplement color. The figure supports `CLM-019`; it does not turn preference into truth.

Long description: Two admissible outputs labeled A and B sit on a blinded comparison bench beside one rubric and a visible disagreement marker.

FIG-014 - V2-F07.2 - Bias can move the label without improving the criterion. Essential labels: length, position, style, familiarity, coupling. Distinct magnet shapes supplement color. The figure supports `CLM-020` and `CLM-021`; it reports no prevalence.

Long description: Length, position, style, familiarity, and coupling magnets pull a preference marker away from the intended rubric target.

FIG-015 - V2-F08.1 - Target gain is bounded by protected behavior. Essential labels: target, retention, multilingual, abstention, control. Shield shapes supplement color. The figure supports `CLM-022` through `CLM-024`; it is not safety certification.

Long description: A target-improvement core is surrounded by retention, multilingual, abstention, and control shields with named owners and stop gates.

FIG-016 - V2-F08.2 - A rising target cannot hide protected decline. Essential labels: target, retention, control, block. Line styles and threshold gates supplement color. The figure supports the forbidden-regression decision; values are illustrative only.

Long description: Across synthetic checkpoints, a target curve rises while retention and control curves fall through blocking thresholds.

FIG-017 - V2-F09.1 - Every run state points back to one manifest. Essential labels: data, config, seed, code, hardware, checkpoint, logs, eval. Connector shapes supplement color. The figure supports `CLM-025`; it does not show a successful run.

Long description: Data, configuration, seeds, code, hardware, checkpoints, logs, and behavioral evaluators connect into one traceable training experiment cockpit.

FIG-018 - V2-F09.2 - Training state is larger than model weights. Essential labels: batch, sequence, weights, activations, gradients, optimizer, checkpoint. Shelf shapes supplement color. The figure supports `CLM-026`; it reports no hardware capacity.

Long description: Batch and sequence flow through weights, activations, gradients, optimizer state, and checkpoint shelves that compete for finite memory.

FIG-019 - V2-F10.1 - Optimization creates candidates; evaluation creates evidence. Essential labels: base, examples, checkpoints, behavior eval, selection. Shape and connector direction supplement color. The figure supports `CLM-028` and `CLM-029`; it depicts a process, not a Mosaic training result.

Long description: A frozen base model receives reviewed supervised examples, produces several checkpoints, and sends each checkpoint to an independent behavior evaluation before a selection gate.

FIG-020 - V2-F10.2 - The best loss can be the wrong checkpoint. Essential labels: train, held-out, retention, stop. Line pattern and marker shapes supplement color. The figure supports `CLM-029` and `CLM-030`; all curves are illustrative.

Long description: Training loss continues downward while held-out behavior flattens and protected retention turns downward, causing an earlier stop marker to win over the final checkpoint.

FIG-021 - V2-F11.1 - The adapter is small; its dependency surface is not. Essential labels: frozen, adapter, base revision, compatibility. Texture and connector shape supplement color. The figure supports `CLM-031`; it makes no quality claim.

Long description: Small trainable low-rank adapter inserts sit beside frozen base-model layers, while a compatibility key binds the adapter to one exact base revision.

FIG-022 - V2-F11.2 - Efficient adaptation lives on a behavior-resource frontier. Essential labels: target, retention, language, memory, storage, runtime. Axis shapes and pass-block symbols supplement color. The figure supports `CLM-032` and `CLM-033`; values are illustrative.

Long description: No-tune, supervised, and two adapter candidates are compared across target behavior, retention, language slices, memory, storage, and runtime compatibility rather than parameter count alone.

FIG-023 - V2-F12.1 - The preference objective remains coupled to data and reference. Essential labels: pairs, policy, reference, objective, candidate, independent eval. Connector direction and gate shape supplement color. The figure supports `CLM-034` and `CLM-035`; it is method-level, not a training result.

Long description: Audited preference pairs feed a policy and frozen reference into a preference objective, producing a candidate that must pass an independent evaluation gate.

FIG-024 - V2-F12.2 - A proxy can reward the visible style and miss the contract. Essential labels: proxy, shortcut, unsupported, abstention, regression, reject. Alarm icons and barrier shapes supplement color. The figure supports `CLM-035` and `CLM-036`; the scene is a failure simulation.

Long description: A verbose confident response takes a shortcut to a high proxy score while unsupported-detail, abstention, and language-regression alarms block promotion.

FIG-025 - V2-F13.1 - Methods answer different failure hypotheses. Essential labels: residual, repair, SFT/PEFT, distill, continue, replace, reject. Branch shapes supplement color. Supports CLM-037 to CLM-039; not a ranking.

Long description: Residual diagnosis routes separately to repair, SFT or PEFT, distillation, continued pretraining, replacement, no project, or rejection.

FIG-026 - V2-F13.2 - Transfer sources remain bounded by lineage and tests. Essential labels: teacher, corpus, provenance, rights, contamination, evaluate. Supports CLM-037 and CLM-038.

Long description: Teacher and corpus paths cross provenance, rights, contamination, and independent evaluation barriers before reaching a student or base model.

FIG-027 - V2-F14.1 - Release identity spans the behavior-facing system. Essential labels: base, tokenizer, template, adapter, config, schema, runtime, license, evidence, hash. Supports CLM-040 and CLM-041.

Long description: One versioned crate contains separately labeled base, tokenizer, template, adapter, configuration, schema, runtime, license, evidence, and hash compartments.

FIG-028 - V2-F14.2 - Integrity cannot substitute for compatibility. Essential labels: match, mismatch, hash, compatibility, stop. Supports CLM-042.

Long description: Matched base, tokenizer, template, adapter, and runtime keys open a lock while a validly hashed mismatched combination stops.

FIG-029 - V2-F15.1 - Weights share memory with growing request state. Essential labels: weights, workspace, activations, KV, adapter, margin, concurrency. Supports CLM-043 and CLM-045.

Long description: Finite shelves hold weights, workspace, activations, KV cache, adapter state, and safety margin while concurrent sequences consume cache blocks.

FIG-030 - V2-F15.2 - Utilization can lengthen a user-visible tail. Essential labels: arrivals, batch, cache, TTFT, throughput, p99 tail, limit. Supports CLM-044; trace is synthetic.

Long description: Short, long, and burst arrivals enter batching and cache scheduling; a larger batch raises throughput while p99 TTFT crosses the interactive limit.

FIG-031 - V2-F16.1 - No single configuration dominates every dimension. Essential labels: behavior, p99 tail, throughput, memory, cost, reject. Marker shapes supplement color. Supports CLM-046 to CLM-048; all points are synthetic.

Long description: Reference, quantized, alternate-runtime, and scheduling candidates occupy different positions across protected behavior, p99 tail latency, throughput, memory, and cost, with unsafe points rejected.

FIG-032 - V2-F16.2 - Pressure changes service state, not hidden truthfulness. Essential labels: full, alternate, context, queue, abstain, fallback, stop. Step shapes supplement color. Supports the operational fallback evidence for CLM-048.

Long description: Runtime pressure follows bounded steps from full qualified service to alternate candidate, reduced optional context, queue, abstain, fallback, and stop.

FIG-033 - V2-F17.1 - Evidence and distributed authority meet at the cohort gate. Essential labels: target, retention, control, runtime, signals, owner, cohort, rollback, hold. Distinct keys supplement color. Supports CLM-049 and CLM-051.

Long description: Target, retention, control, runtime, privacy-minimized observability, owner decisions, cohort plan, and rollback evidence enter a release gate that can issue hold.

FIG-034 - V2-F17.2 - Change is complete only when evidence and controls replay. Essential labels: detect, contain, diagnose, correct, verify, replay, decide, rollback, deprecate, learn. Direction and state shapes supplement color. Supports CLM-050 and CLM-051.

Long description: A controlled loop moves through detect, contain, diagnose by layer, correct, verify, replay, authority decision, rollback or ramp, deprecate, and learn.

Sources

SRC-001 - Ashish Vaswani et al.. Attention Is All You Need. arXiv. https://arxiv.org/abs/1706.03762 (accessed 2026-08-16).

SRC-002 - Taku Kudo, John Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. arXiv. https://arxiv.org/abs/1808.06226 (accessed 2026-08-16).

SRC-003 - Rico Sennrich, Barry Haddow, Alexandra Birch. Neural Machine Translation of Rare Words with Subword Units. arXiv. https://arxiv.org/abs/1508.07909 (accessed 2026-08-16).

SRC-007 - OpenAI Developers. Model optimization. OpenAI. https://developers.openai.com/api/docs/guides/model-optimization (accessed 2026-08-16).

SRC-009 - Percy Liang et al.. Holistic Evaluation of Language Models. Stanford CRFM / arXiv. https://arxiv.org/abs/2211.09110 (accessed 2026-08-16).

SRC-010 - OpenAI Developers. Working with evals. OpenAI. https://developers.openai.com/api/docs/guides/evals (accessed 2026-08-16).

SRC-011 - Anthropic documentation team. Define success criteria and build evaluations. Anthropic. https://platform.claude.com/docs/en/test-and-evaluate/develop-tests (accessed 2026-08-16).

SRC-015 - Hugging Face Transformers maintainers. Writing a chat template. Hugging Face. https://huggingface.co/docs/transformers/en/chat_templating_writing (accessed 2026-08-16).

SRC-016 - Nelson F. Liu et al.. Lost in the Middle: How Language Models Use Long Contexts. arXiv. https://arxiv.org/abs/2307.03172 (accessed 2026-08-16).

SRC-019 - Patrick Lewis et al.. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. arXiv. https://arxiv.org/abs/2005.11401 (accessed 2026-08-16).

SRC-025 - Shahul Es et al.. RAGAS: Automated Evaluation of Retrieval Augmented Generation. arXiv. https://arxiv.org/abs/2309.15217 (accessed 2026-08-16).

SRC-037 - Ahmet Üstün et al.. Aya Model: An Instruction Finetuned Open-Access Multilingual Language Model. Cohere For AI / arXiv. https://arxiv.org/abs/2402.07827 (accessed 2026-08-16).

SRC-039 - Nathan Lambert et al.. Tülu 3: Pushing Frontiers in Open Language Model Post-Training. Allen Institute for AI / arXiv. https://arxiv.org/abs/2411.15124 (accessed 2026-08-16).

SRC-041 - Edward J. Hu et al.. LoRA: Low-Rank Adaptation of Large Language Models. arXiv. https://arxiv.org/abs/2106.09685 (accessed 2026-08-16).

SRC-042 - Hugging Face PEFT maintainers. Parameter-efficient fine-tuning methods. Hugging Face. https://huggingface.co/docs/peft/main/methods/overview (accessed 2026-08-16).

SRC-043 - Tim Dettmers et al.. QLoRA: Efficient Finetuning of Quantized LLMs. arXiv. https://arxiv.org/abs/2305.14314 (accessed 2026-08-16).

SRC-044 - Rafael Rafailov et al.. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. arXiv. https://arxiv.org/abs/2305.18290 (accessed 2026-08-16).

SRC-046 - Geoffrey Hinton, Oriol Vinyals, Jeff Dean. Distilling the Knowledge in a Neural Network. arXiv. https://arxiv.org/abs/1503.02531 (accessed 2026-08-16).

SRC-048 - Suchin Gururangan et al.. Don't Stop Pretraining: Adapt Language Models to Domains and Tasks. arXiv. https://arxiv.org/abs/2004.10964 (accessed 2026-08-16).

SRC-051 - PyTorch maintainers. Automatic Mixed Precision. PyTorch. https://docs.pytorch.org/docs/stable/accelerator/amp.html (accessed 2026-08-16).

SRC-054 - Tri Dao et al.. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. arXiv. https://arxiv.org/abs/2205.14135 (accessed 2026-08-16).

SRC-006 - Long Ouyang et al.. Training language models to follow instructions with human feedback. arXiv. https://arxiv.org/abs/2203.02155 (accessed 2026-08-16).

SRC-008 - National Institute of Standards and Technology. Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile. NIST. https://doi.org/10.6028/NIST.AI.600-1 (accessed 2026-08-16).

SRC-034 - Luca Soldaini et al.. Dolma: an Open Corpus of Three Trillion Tokens for Language Model Pretraining Research. Allen Institute for AI / arXiv. https://arxiv.org/abs/2402.00159 (accessed 2026-08-16).

SRC-035 - Katherine Lee et al.. Deduplicating Training Data Makes Language Models Better. arXiv. https://arxiv.org/abs/2107.06499 (accessed 2026-08-16).

SRC-036 - Timnit Gebru et al.. Datasheets for Datasets. arXiv. https://arxiv.org/abs/1803.09010 (accessed 2026-08-16).

SRC-038 - Yizhong Wang et al.. Self-Instruct: Aligning Language Models with Self-Generated Instructions. arXiv. https://arxiv.org/abs/2212.10560 (accessed 2026-08-16).

SRC-040 - Hugging Face TRL maintainers. SFT Trainer. Hugging Face. https://huggingface.co/docs/trl/sft_trainer (accessed 2026-08-16).

SRC-018 - National Institute of Standards and Technology. Adversarial Machine Learning: A Taxonomy and Terminology of Attacks and Mitigations. NIST. https://doi.org/10.6028/NIST.AI.100-2e2025 (accessed 2026-08-16).

SRC-028 - Lianmin Zheng et al.. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. arXiv. https://arxiv.org/abs/2306.05685 (accessed 2026-08-16).

SRC-030 - Nathan Lambert et al.. RewardBench: Evaluating Reward Models for Language Modeling. arXiv. https://arxiv.org/abs/2403.13787 (accessed 2026-08-16).

SRC-049 - Ilya Loshchilov, Frank Hutter. Decoupled Weight Decay Regularization. arXiv. https://arxiv.org/abs/1711.05101 (accessed 2026-08-16).

SRC-050 - PyTorch maintainers. FullyShardedDataParallel. PyTorch. https://docs.pytorch.org/docs/stable/fsdp.html (accessed 2026-08-16).

SRC-065 - OpenAI Developers. Safety best practices. OpenAI. https://developers.openai.com/api/docs/guides/safety-best-practices (accessed 2026-08-16).

SRC-027 - Google Cloud. Evaluate a judge model. Google Cloud. https://docs.cloud.google.com/gemini-enterprise-agent-platform/models/evaluate-judge-model (accessed 2026-08-17).

SRC-029 - Yang Liu et al.. G-Eval: NLG Evaluation using GPT-4 with Better Human Alignment. arXiv. https://arxiv.org/abs/2303.16634 (accessed 2026-08-16).

SRC-033 - OpenAI Developers. Graders. OpenAI. https://developers.openai.com/api/docs/guides/graders (accessed 2026-08-16).

SRC-045 - Hugging Face TRL maintainers. DPO Trainer. Hugging Face. https://huggingface.co/docs/trl/dpo_trainer (accessed 2026-08-16).

SRC-061 - Hugging Face Transformers maintainers. Bitsandbytes quantization. Hugging Face. https://huggingface.co/docs/transformers/quantization/bitsandbytes (accessed 2026-08-16).

SRC-005 - Jordan Hoffmann et al.. Training Compute-Optimal Large Language Models. arXiv. https://arxiv.org/abs/2203.15556 (accessed 2026-08-16).

SRC-013 - Google AI for Developers. Structured outputs. Google. https://ai.google.dev/gemini-api/docs/structured-output (accessed 2026-08-16).

SRC-047 - Victor Sanh et al.. DistilBERT, a distilled version of BERT. arXiv. https://arxiv.org/abs/1910.01108 (accessed 2026-08-16).

SRC-052 - Margaret Mitchell et al.. Model Cards for Model Reporting. arXiv. https://arxiv.org/abs/1810.03993 (accessed 2026-08-16).

SRC-053 - Hugging Face. Safetensors. Hugging Face. https://huggingface.co/docs/safetensors/index (accessed 2026-08-16).

SRC-055 - Woosuk Kwon et al.. Efficient Memory Management for Large Language Model Serving with PagedAttention. arXiv. https://arxiv.org/abs/2309.06180 (accessed 2026-08-16).

SRC-058 - Yaniv Leviathan et al.. Accelerating Large Language Model Decoding with Speculative Sampling. arXiv. https://arxiv.org/abs/2302.01318 (accessed 2026-08-16).

SRC-059 - NVIDIA. Quantization in TensorRT-LLM. NVIDIA. https://nvidia.github.io/TensorRT-LLM/latest/features/quantization.html (accessed 2026-08-16).

SRC-060 - NVIDIA. TensorRT-LLM Benchmarking. NVIDIA. https://nvidia.github.io/TensorRT-LLM/developer-guide/perf-benchmarking.html (accessed 2026-08-16).

SRC-062 - vLLM Project. vLLM documentation. vLLM Project. https://docs.vllm.ai/en/latest/ (accessed 2026-08-16).

SRC-012 - Google AI for Developers. Prompt design strategies. Google. https://ai.google.dev/gemini-api/docs/prompting-strategies (accessed 2026-08-16).

SRC-014 - Hugging Face Transformers maintainers. Text generation strategies. Hugging Face. https://huggingface.co/docs/transformers/main/generation_strategies (accessed 2026-08-16).

SRC-031 - NIST. 2024 NIST GenAI Pilot Study: Text-to-Text Evaluation Overview and Results. NIST. https://doi.org/10.6028/NIST.AI.700-1 (accessed 2026-08-16).

SRC-032 - Anthropic. Demystifying evals for AI agents. Anthropic. https://www.anthropic.com/engineering/demystifying-evals-for-ai-agents (accessed 2026-08-16).

SRC-056 - Ji Lin et al.. AWQ: Activation-aware Weight Quantization for LLM Compression and Acceleration. arXiv. https://arxiv.org/abs/2306.00978 (accessed 2026-08-16).

SRC-057 - Elias Frantar et al.. GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers. arXiv. https://arxiv.org/abs/2210.17323 (accessed 2026-08-16).

SRC-063 - OpenAI Developers. Latency optimization. OpenAI. https://developers.openai.com/api/docs/guides/latency-optimization (accessed 2026-08-16).

SRC-064 - OpenAI Developers. Deprecations. OpenAI. https://developers.openai.com/api/docs/deprecations (accessed 2026-08-16).

Edition record

Version: 1.0.0

Published: 2026-08-17

Canonical URL: https://komalnakrani.com/books/llm-adaptation-and-runtime/

Recorded errata: 0

Chapters: 17

Figures: 34

Sources: 57