✦ PLAYBOOK · COST OPTIMIZATION

Model Routing

Sending each request to the right model by task, difficulty and cost — rules vs classifiers, cascades with fallback, evaluating routes, provider redundancy and gateway design.

FORMAT	LEVEL	READ TIME	UPDATED
Playbook	Advanced	14 min	Oct 2026

OVERVIEW

What this playbook covers

Using one large model for everything is simple and expensive; using one small model for everything is cheap and fails on the hard cases. Model routing sits between them: a decision layer that sends each request to the cheapest model that can handle it well, falls back when it cannot, and survives provider outages. This playbook covers routing strategies, how to build and evaluate a router, cascade design, multi-provider redundancy and the gateway that ties it together.

KEY TAKEAWAYS

- Route by task type first — it is cheap, explainable and captures much of the benefit before any learned router.
- Add difficulty-based routing or cascades only where evals show a small model handles a meaningful share of traffic well.
- A cascade needs a reliable signal of failure — validation, confidence or a verifier — or it just adds latency.
- Evaluate the whole routed system against the single-best-model baseline on quality, cost and latency.
- Keep at least one fallback route per critical capability on a different provider or deployment.
- Centralize routing in a gateway so policy, metering, retries and failover live in one place.

WHO IT'S FOR

AI platform engineers

Engineering leads owning model spend

SRE and reliability teams

Architects choosing model providers

HOW TO USE IT

Start with section 01 to pick a routing strategy, use 02–03 to build and evaluate it, and design the gateway with sections 04–05. The checklist in 06 is the launch gate.

CONTENTS

01 Routing strategies
02 Rules vs classifiers
03 Cascades with fallback
04 Evaluating routes
05 Provider redundancy and gateway design
06 Launch checklist

01 / STRATEGIES

Routing strategies

Routing decides which model serves a request. Strategies differ in how much they know about the request and how costly the decision itself is.

STRATEGY	DECIDES ON	STRENGTHS	WEAKNESSES
Static by task	Feature or endpoint	Simple, explainable, zero overhead	Ignores variation within a task
Rules	Input length, language, tenant tier, tools needed	Predictable; easy to audit	Brittle; rules accumulate
Classifier router	Predicted difficulty or category	Adapts to request content	Needs labelled data and its own evals
Cascade	Outcome of a cheaper attempt	Pays for large models only when needed	Adds latency on escalated requests
Cost / latency aware	Budget, SLA, current provider health	Meets constraints dynamically	Harder to reason about quality

SMALL / FAST

Mechanical tasks

Classification, extraction, routing, short rewrites, query expansion and guardrail checks.

MID-TIER

Grounded generation

RAG answers, summaries and drafting where context does most of the work.

FRONTIER

Hard reasoning

Multi-step planning, complex code, ambiguous analysis and agent orchestration.

✦ 02 / ROUTERS

Rules vs classifiers

Start with rules; move to a learned router only when you have the data to train and evaluate it.

RULE-BASED ROUTING	CLASSIFIER ROUTING
Route on task, length, language, tier	Predicts which model will succeed
Deterministic and testable	Trained on labelled outcomes per model
No training data needed	Captures subtle difficulty signals
Misses hard requests that look simple	Must be monitored for drift like any model

Building a classifier router

01 **Collect representative requests**
Sample real traffic for the task, stratified by intent and difficulty.

02 **Run every candidate model**
Score each model's output on each request with your eval suite to produce labels.

03 **Label the cheapest adequate model**
For each request, record the cheapest model whose output passed the quality bar.

04 **Train a lightweight router**
A small classifier on embeddings and request features keeps routing overhead low.

05 **Bias toward safety**
Set the decision threshold so uncertain requests go to the stronger model.

✦ 03 / CASCADES

Cascades with fallback

A cascade tries a cheaper model first and escalates when the result fails a check. It works only if failure is detectable.

01 Call small model → 02 Validate output → 03 Pass: return → 04 Fail: call larger model → 05 Still failing: fallback or human

Escalation signals

SIGNAL	WORKS WELL FOR	CAVEAT
Schema / parser validation	Structured output and tool calls	Catches format errors, not wrong content
Deterministic business checks	Extraction with known constraints	Only covers rules you can write
Self-reported confidence	Classification with calibrated scores	Generative models are often overconfident
Verifier model	Grounded answers, math, code with tests	Adds a call; the verifier needs its own evals
Explicit refusal or “unsure” output	Tasks where the model can abstain	Must prompt for and allow abstention

✦ FIELD NOTE

Watch the escalated-path latency

Escalated requests pay for both calls. If a large share of requests escalate, the cascade is slower and may be no cheaper than routing them straight to the larger model.

04 / EVALUATION

Evaluating routes

A router is judged against a baseline: the single best model for every request. The question is how much cost and latency you save for how much quality.

01

Quality vs baseline

Run the golden set through the routed system and through the strongest model alone; compare per metric and slice.

02

Routing accuracy

How often hard requests were sent to a weak model (costly errors) and easy ones to a strong model (wasted spend).

03

Cost and latency mix

Share of traffic per route, average cost per request and p95 latency including escalations.

04

Slice safety

Check critical slices — safety, regulated topics, high-value customers — separately; never let averages hide them.

In production

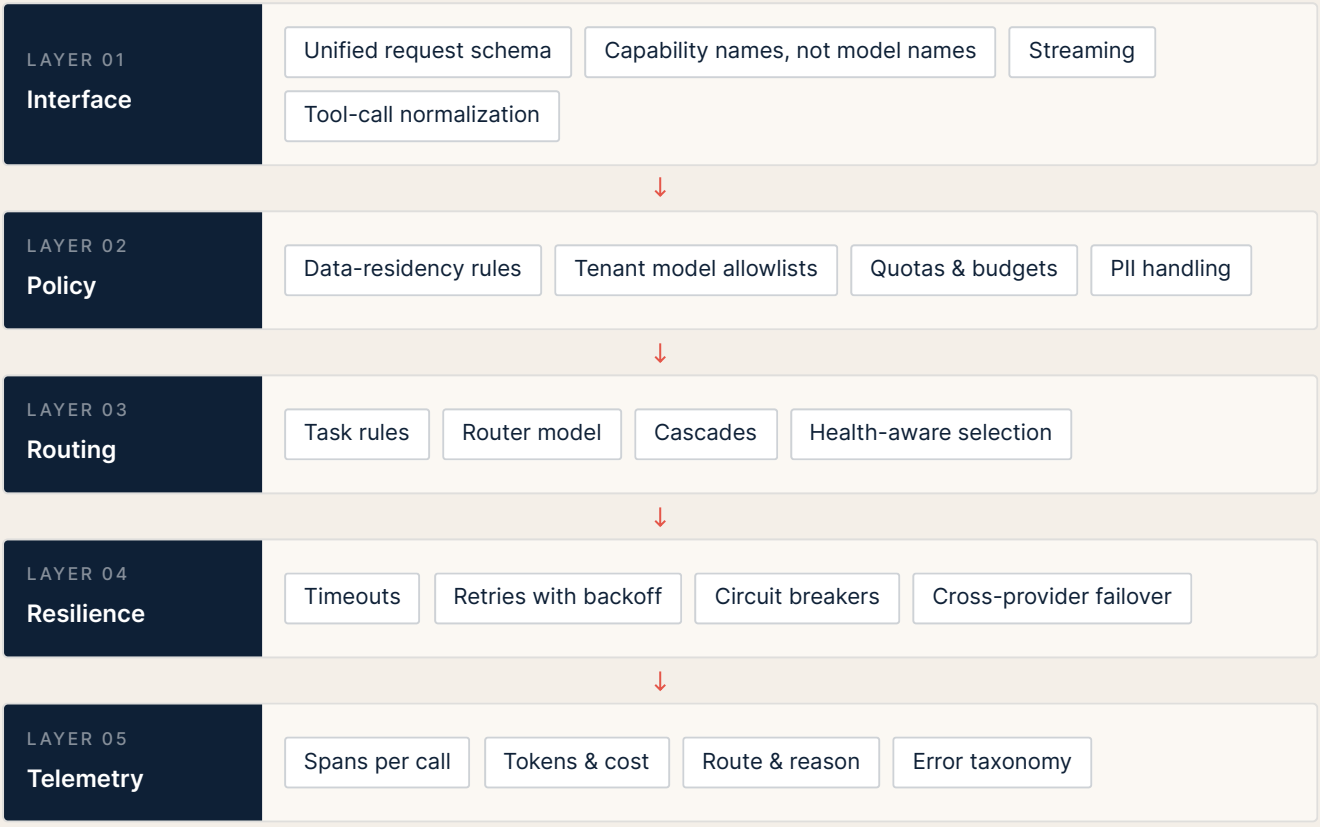
- Log the route chosen, the reason and the outcome on every request's trace.
- Shadow-test new routes on a sample of traffic before switching them on.
- Re-run routing evals when any model in the pool changes version — the cheapest adequate model can change.

> **Route on evidence. The right model is the cheapest one your evals say is good enough — and a second one ready when it is not there.**
>
> SPLENDENSLABS ENGINEERING PRINCIPLE

✦ 05 / GATEWAY

Provider redundancy and gateway design

A model gateway is the single entry point for every model call. It is where routing, failover, quotas and metering belong.



Designing for provider redundancy

01 Name capabilities, not models

Callers ask for `summarize-fast` or `reason-strong`; the gateway maps capabilities to models and can change the mapping without code changes.

02 Qualify a second option

For each critical capability, evaluate a fallback model on a different provider or deployment against the same eval suite, with its own versioned prompt variant.

03 Fail over on health, not hope

Circuit breakers trip on error rate and latency, route to the fallback and probe the primary before restoring.

✦ 06 / OPERATIONS

Launch checklist

Use this before turning on routing for a production feature.

STRATEGY

- [ ] Route by task defined for every feature
- [ ] Cheapest adequate model chosen per task from eval results
- [ ] Uncertain requests default to the stronger model

EVALUATION

- [ ] Routed system compared to single-best-model baseline
- [ ] Critical slices checked separately
- [ ] Routing evals re-run on every model version change

RESILIENCE

- [ ] Fallback route qualified for each critical capability
- [ ] Circuit breakers and timeouts configured
- [ ] Failover tested by forcing a provider outage

GATEWAY

- [ ] All model calls go through the gateway
- [ ] Route, reason, tokens and cost on every trace
- [ ] Per-tenant quotas and allowlists enforced

GET IN TOUCH

Let's Talk

We design model gateways and routing layers that cut spend without cutting quality, and keep working when a provider does not. Bring us your model mix and your traffic.

- Routing strategy and router evaluation
- Cascade and verifier design
- Multi-provider gateway and failover

BOOK AN AI STRATEGY SESSION →

EMAIL
hello@splendenslabs.com

WEB
splendenslabs.com

OFFICE
Sona Incubation Foundation, Salem