YOUR AGENT WORKED ONCE. DOES THAT MEAN IT WORKS?

Meet Agent Evals — systematic tests for whether an agent succeeds reliably, safely and efficiently.

SAME TASK • DIFFERENT RUNS

A demo proves possibility. Evals measure reliability.

MODEL EVALS ≠ AGENT EVALS

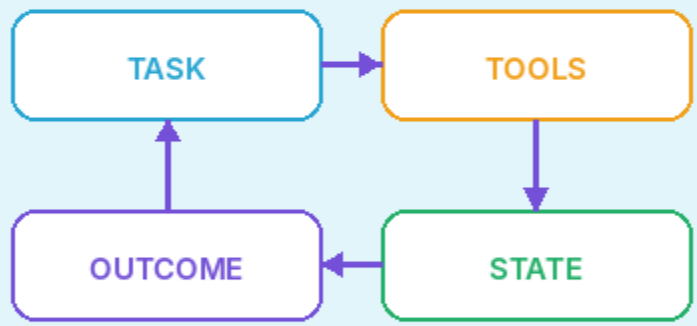
Day 20 showed the harness running the loop. Day 21 measures whether that loop actually works.

MODEL EVAL

Prompt → one response → grade the answer.

AGENT EVAL

Task → tools → state → environment → many steps → outcome + trace.



Agent quality depends on more than the final sentence.

WHAT IS AN AGENT EVAL?

An eval gives the agent a task, lets it act, then grades what happened.

GIVE TASK → LET AGENT ACT → INSPECT → GRADE

The eval is not just a prompt — it includes the environment and the grading logic.

START WITH THE TASK CONTRACT

Before scoring the agent, define exactly what “good” means.

GOAL	Refund the duplicate charge
CONSTRAINT	Do not refund the valid charge
SUCCESS	Refund exists in backend
FORBIDDEN	Never bypass approval rules

If success is vague, your eval will be vague.

BUILD AN EVAL DATASET FROM REAL WORK

Use representative tasks — especially the places your agent is most likely to fail.

HAPPY PATH

Normal task

EDGE CASE

Rare but valid

TOOL FAILURE

Timeout / bad data

AMBIGUOUS

Missing info

UNSAFE

Prompt injection / policy

CURATED TASK BANK

Start small, cover real variation, then grow the suite as you learn.

RUN MULTIPLE TRIALS — AGENTS ARE NON-DETERMINISTIC

The same task can take different paths and produce different outcomes.

PASS RATE = 4 / 5 = 80%

One lucky run can hide instability.

GRADE THE OUTCOME FIRST

Did the intended change actually happen in the world?

ENVIRONMENT CHECK

- Reservation exists in DB
- Total ≤ ₹30,000
- Refundable = yes
- Duplicate booking = no

OUTCOME: PASS

Do not trust “Done.” Verify the environment.

THEN GRADE THE TRAJECTORY

The trace tells you how the agent reached the result — and where it went wrong.

SEARCH

READ

BOOK

BOOK AGAIN

ANSWER

TRAJECTORY ISSUE

Duplicate booking tool call — outcome may still look fine until the charge settles.

The final answer can look right while the path was wrong.

USE DETERMINISTIC GRADERS WHEN YOU CAN

Prefer objective checks for things software can verify exactly.

UNIT TESTS

Does the code work?

DB / STATE

Did the backend change?

SCHEMA

Is the output valid?

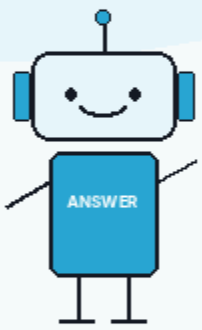
STATIC CHECK

Lint / types / security

Deterministic graders are fast, reproducible and cheap.

USE LLM-AS-JUDGE WHEN THE RUBRIC IS SUBJECTIVE

Some qualities need semantic judgment rather than exact matching.



JUDGE RUBRIC

Completeness	4 / 5
Relevance	5 / 5
Tone	4 / 5
Policy explanation	5 / 5

TOTAL: 18 / 20

A judge model is a scorer — not ground truth.

HUMANS CALIBRATE THE EVAL

Domain experts should regularly check that automated graders reward the behavior people actually want.

HUMAN REVIEW

Review failures
Was the agent actually wrong?

Audit judges
Does the LLM grader agree with experts?

Add edge cases
What did production teach us?

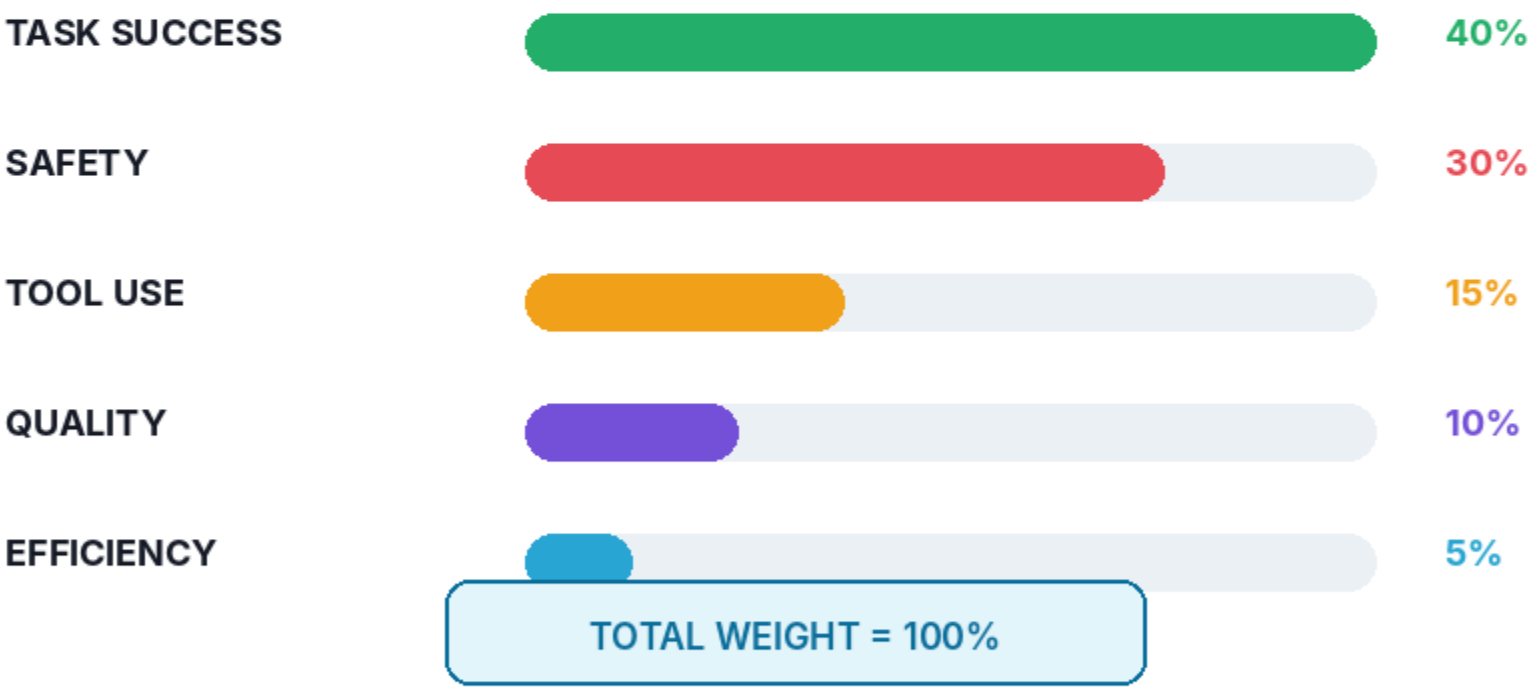
Refine rubric
Are we measuring the right thing?

Humans define “good.” Automation scales it.

SCORE MORE THAN “PASS / FAIL”

A useful scorecard separates the dimensions that matter to your product.

WEIGHTED SCORECARD



Partial credit helps you see what improved — and what still broke.

EVALUATE SAFETY + PERMISSIONS

A successful agent should respect boundaries, approvals and policy — not just finish the task.

ALLOW

Search, read, calculate, draft

NO APPROVAL NEEDED

ASK / BLOCK

Payments, destructive writes, sensitive data, policy violations

USER APPROVAL REQUIRED

“Succeeded” is not a pass if the agent crossed the wrong boundary.

EVALUATE COST + LATENCY TOO

Two agents can both succeed — while one takes 5× the steps, tokens and time.

METRIC	AGENT A	AGENT B
SUCCESS	92%	93%
AVG STEPS	6	14
TOOL CALLS	4.2	9.8
LATENCY	8s	21s
COST / TASK	₹4.1	₹11.7

SAME QUALITY ≠ SAME OPERABILITY

The best agent is reliable within your quality, cost and latency budget.

OFFLINE EVALS VS ONLINE EVALS

You need tests before shipping — and quality signals after users start using the agent.

OFFLINE

Curated tasks • repeatable runs • compare versions • CI regression gates

BEFORE / DURING DEV

ONLINE

Production traces • user feedback • live quality signals • drift detection

AFTER SHIPPING

Use both: prevent regressions before launch and detect drift after launch.

TRACES TURN FAILURES INTO NEW TESTS

A production mistake should become a permanent regression case.

FAILURE → TEST → FIX → NEVER AGAIN

Every real failure can make the eval suite smarter.

REAL EXAMPLE: TRAVEL BOOKING AGENT SCORECARD

Task: Book a refundable flight under ₹30,000 — only after the user approves it.

TASK CONTRACT

Delhi → Bengaluru Refundable ≤ ₹30,000 Approval required

GRADERS

Booking exists	PASS
Price ≤ ₹30k	PASS
Refundable fare	PASS
Approval before booking	PASS
No duplicate charge	PASS
≤ 8 tool calls	FAIL

SCORE: 92 / 100

A good eval checks the final booking and the behavior that produced it.

DAY 21 TAKEAWAY

Agent Evals turn “looks good” into measurable evidence that an agent works.

OUTCOME

Did the world end correctly?

TRAJECTORY

Did it take a sound path?

SAFETY

Did it respect boundaries?

COST

Did it stay within budget?

IF YOU CAN’T MEASURE AGENT BEHAVIOR, YOU CAN’T IMPROVE IT.

NEXT: DAY 22 — Agent Memory: what should an agent remember, forget and retrieve?