How to Compare LangSmith, Phoenix, Weave, and Foundry as Evaluation Substrates

A monetization play: sell “evaluation substrate selection” as a paid decision, then monetize the implementation, migrations, and a recurring agent QA monitoring retainer.

Source https://yetyield.com/blog/compare-langsmith-phoenix-weave-foundry-as-eval-substrates/

If you want evaluation to become yield, you need more than “a rubric” or “a dataset”.

You need an evaluation substrate that makes it easy to ship three things repeatedly:
• a gate (block bad changes)
• a scorecard (explain what broke)
• a cadence (make it recurring)

The monetization angle

“Which eval platform should we use?” is not a tooling question.

It is a paid decision because the wrong substrate creates recurring cost in all the places buyers feel pain:
• migrations when APIs change
• evaluator drift (rubrics get stale)
• datasets that don’t match production
• costs that silently explode

Your paid ladder:
1.Substrate selection audit (fixed scope)
2.Gate + scorecard implementation sprint
3.Ongoing regression monitoring retainer (monthly yield)

This connects directly to the existing YetYield cluster:
• Start with the artifact: How to Turn Agent Evaluation Checklists Into a Paid Product (/blog/how-to-turn-agent-evaluation-checklists-into-a-paid-product/)
• Monetize continuity: Agent Regression Tests Can Be a Retainer Business (/blog/agent-regression-tests-as-a-retainer-business/)

The four substrates (what matters in practice)

The point is not to “rank tools”.

The point is to map tool surfaces to sellable deliverables.

Substrate	Best for	Sellable deliverable	Official entry point
LangSmith	Offline + online evaluation as a workflow, with production monitoring semantics	“Define evaluators + sampling + thresholds + monthly quality scorecard”	https://docs.langchain.com/langsmith/evaluation
Phoenix	Flexible evaluators with SDK and a server-side UI, including LLM-as-judge patterns	“Evaluator pack for RAG/agent traces + a repeatable eval pipeline”	https://arize.com/docs/phoenix/evaluation/llm-evals
W&B Weave	Evaluation with scorers/judges + tracing + cost tracking, integrated with W&B workflows	“Cost-aware QA scorecards + regression analysis on model versions”	https://docs.wandb.ai/models/tables/evaluate-models
Azure AI Foundry	Portal-driven evaluations across AI quality + NLP metrics + risk/safety metrics	“Enterprise readiness audit: quality + risk evidence pack”	https://learn.microsoft.com/en-us/azure/ai-foundry/how-to/evaluate-generative-ai-app

How to choose without getting stuck

Use these four decision tests.

Test 1: Where will your dataset come from?

If you can’t continuously harvest cases from production traces, your evaluation “system” is just a one-time report.

Phoenix explicitly supports running evaluations on traces, experiment results, or any dataset, using both deterministic evaluators and LLM-as-judge evaluators.
Official anchor: https://arize.com/docs/phoenix/evaluation/llm-evals

Test 2: Can you run a true gate, not just a dashboard?

A gate needs:
• thresholds
• re-run policy for variance
• a pass/fail decision surfaced in CI

If you need a gate blueprint, start here:
• How to Use Vertex GenAI Evaluation Service as an Agent QA Gate (/blog/vertex-genai-evaluation-service-as-agent-qa-gate/)

Test 3: Do you have “online evaluation” semantics?

The retainer is usually funded by “we keep quality stable as you change things.”

That requires:
• sampling rules (evaluate 10% of runs + 100% of failures)
• evaluator drift management
• incident response loops

If your buyer is already in the LangChain ecosystem, LangSmith’s evaluation surface is a clean substrate for this kind of retainer packaging.
Official anchor: https://docs.langchain.com/langsmith/evaluation

Test 4: Can you price the cost boundary?

Evaluation is not free.

You need to productize:
• cost budgets (what you will spend per month)
• concurrency/rate-limit handling (so runs don’t fail and waste budget)
• reporting that ties cost to quality improvements

W&B Weave explicitly positions evaluation with “cost tracking” alongside scorers/judges and tracing, which maps cleanly to this offer framing.
Official anchor: https://docs.wandb.ai/models/tables/evaluate-models

The “substrate audit” deliverables (what clients pay for)

Deliverable 1: Substrate decision memo
• current stack constraints (cloud, vendor lock-in, existing observability)
• which evaluation surfaces exist today (gate vs dashboard vs portal)
• migration risk summary (what changes will break you)
• recommended substrate + why

Deliverable 2: Evaluation architecture spec
• dataset schema + case harvesting plan
• evaluator library (rubrics, deterministic checks, LLM judges)
• thresholds + escalation rules
• reporting format (scorecard)

Deliverable 3: First-month scorecard + fix sprint
• baseline run
• top failure modes ranked by business impact

• scoped fixes (prompt/tool schema/guardrails)

This is how “evaluation platforms” become yield.

Next research direction: create platform-specific playbooks for each substrate (LangSmith vs Phoenix vs Weave vs Foundry) so readers can pick a stack and immediately package a paid offer around it.