

Building Reusable Evaluation Frameworks for Agentic AI Products

Susan Shu Chang
Principal Data Scientist @ Elastic

QCon AI Boston: June 02, 2026

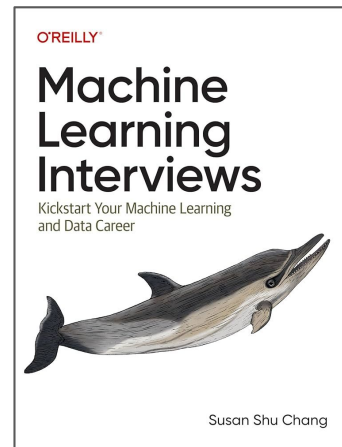


Hi!



Susan Shu Chang

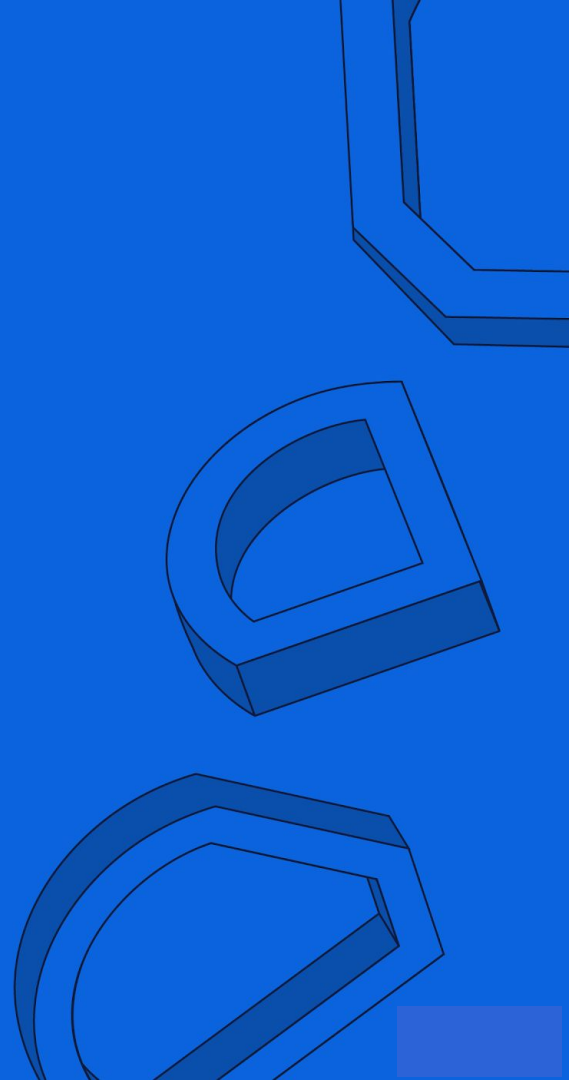
- Principal Data Scientist at Elastic
- Author, Machine Learning Interviews (O'Reilly)
- Speaker at conferences worldwide



Agenda

1. AI agents we're running in production
2. Tracing your application
3. Shared evaluation framework: Overview
4. Shared evaluation framework: Building blocks
5. Tying it all together + Takeaways

**AI agents we're running
in production**



Web & app search / vector search / log search...

Setting the stage on the types of AI Agents we build

The image shows a screenshot of a web application interface. At the top left, there is a logo consisting of three orange vertical bars of increasing height, followed by the word "Blog". To the right of the logo is a search bar with a magnifying glass icon and the text "Find som...". Below the logo and search bar is a navigation menu with the following items: "Everything", "Productivity", "Career Advice" (highlighted with a grey background), "AI/ML", "Open Source", "Business Hub", and "Company".

The main content area features a large heading: "Ask like a human: Implementing semantic search on Stack Overflow". Above the heading is a date icon and the text "JULY 31, 2023". Below the heading is a sub-heading: "Semantic search of a rigid syntax".

At the bottom of the screenshot, there is a section titled "Engineering" with a navigation menu containing "Overview", "Backend", "Culture", "Data / ML", and "Mobile". Below this section is another heading: "Engineering Uber Predictions in Real Time with ELK".

Using Elasticsearch queries, we can quickly see every action the user has done - this is a great way to see whether an account has been stolen, hijacked, or whether the user has done something naughty.

– Tim Pease, Operations Engineer, GitHub

Engineering

Tinder's migration to Elasticsearch 8

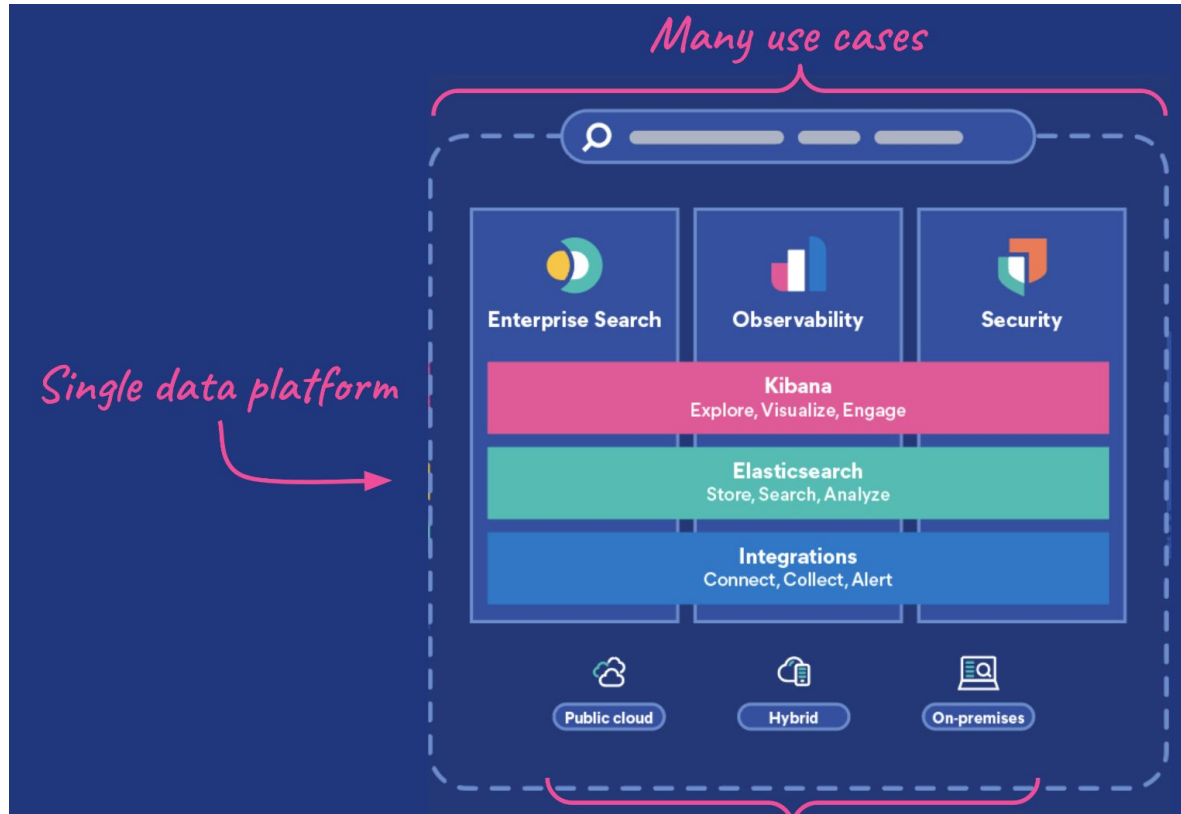
Posted on: April 1, 2025

Share this post



Users ship logs to Elastic / Elastic Security / Observability

Bottom line: Search / RAG to store and extract signals from data



[Security] Attack Discovery: Given logs, find possible attacks

BPFDoor Linux backdoor deployment Open Not shared Attack chain: ○○○○○○●○○○○○○○ Alerts: 4 Take action

Jul 15, 2025 @ 21:16:12.741 Created by: elastic

Backdoor on `SRVNIX05` by `root` View in AI Assistant

Attack discovery Alerts

Summary

On `SRVNIX05`, BPFDoor backdoor is extracted, copied, and executed by `root`.

Details

- On `SRVNIX05`, a user `root` extracts a file (`unzip`) resulting in `74ef6cc38f5a1a80148752b63c117e6846984debd2af806c65887195a8eccc56` in `/home/ubuntu/74ef6cc38f5a1a80148752b63c117e6846984debd2af806c65887195a8eccc56`.
- A bash shell (`bash`) is used to copy this file to `/dev/shm/kdmtmpflush` and execute it with elevated permissions (`sh -c /bin/rm -f /dev/shm/kdmtmpflush;/bin/cp ./74ef6cc38f5a1a80148752b63c117e6846984debd2af806c65887195a8eccc56 /dev/shm/kdmtmpflush && /bin/chmod 755 /dev/shm/kdmtmpflush && /dev/shm/kdmtmpflush`).
- The file is detected as `Linux.Trojan.BPFDoor`, a known Linux backdoor, indicating successful deployment and execution of a persistent threat.

Justification for separation:

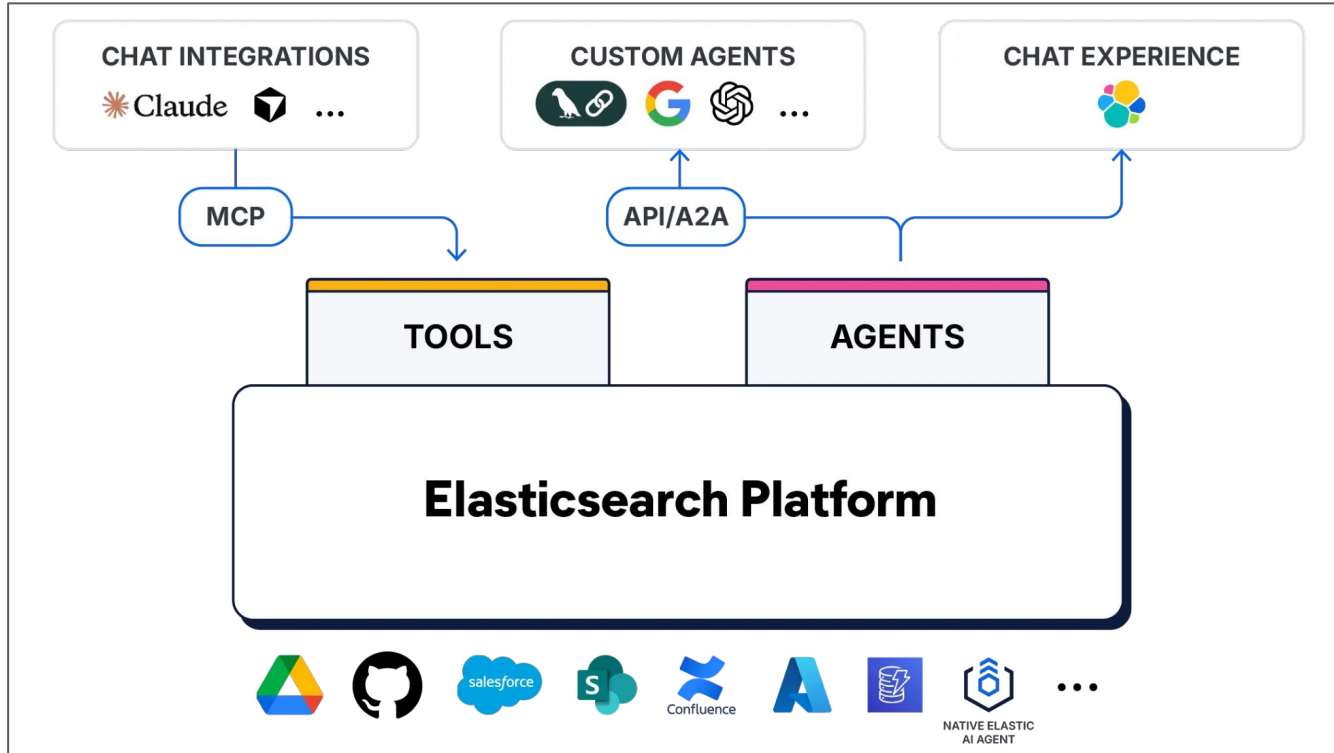
- All alerts are tightly linked by the same host (`SRVNIX05`) and user (`root`), with a clear sequence from file extraction to execution and detection of BPFDoor. No evidence links this chain to other hosts or campaigns.

Attack Chain

Reconnaissance Resource Development **Initial Access** **Execution** **Persistence** Privilege Escalation **Defense Evasion** Credential Access

View in AI Assistant Investigate in Timeline

Elastic Agent Builder (Enterprise chatbot creator): Custom agents on top of existing data



Key questions before shipping changes to production

How do we estimate the quality of responses of these AI applications?

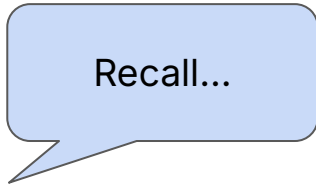
If we make a change, how do we test for regressions?

How can we easily test these results in a repeatable manner?

What we'd build for each AI Agent suite...

1. Test dataset creation
2. Develop LLM- and rules-based evaluators
3. Capture evaluation data (in different places)

[Security] Attack Discovery: Given logs, find possible attacks



BPFDoor Linux backdoor deployment Open Not shared Attack chain: ○○○○○○●○○○○○○○ Alerts: 4 Take action

Jul 15, 2025 @ 21:16:12.741 Created by: elastic

Backdoor on `SRVNIX05` by `root` View in AI Assistant

Attack discovery Alerts

Summary

On `SRVNIX05`, BPFDoor backdoor is extracted, copied, and executed by `root`.

Details

- On `SRVNIX05`, a user `root` extracts a file (`unzip`) resulting in `74ef6cc38f5a1a80148752b63c117e6846984debd2af806c65887195a8eccc56` in `/home/ubuntu/74ef6cc38f5a1a80148752b63c117e6846984debd2af806c65887195a8eccc56`.
- A bash shell (`bash`) is used to copy this file to `/dev/shm/kdmtmpflush` and execute it with elevated permissions (`sh -c /bin/rm -f /dev/shm/kdmtmpflush/bin/cp ./74ef6cc38f5a1a80148752b63c117e6846984debd2af806c65887195a8eccc56 /dev/shm/kdmtmpflush && /bin/chmod 755 /dev/shm/kdmtmpflush && /dev/shm/kdmtmpflush`).
- The file is detected as `Linux.Trojan.BPFDoor`, a known Linux backdoor, indicating successful deployment and execution of a persistent threat.

Justification for separation:

- All alerts are tightly linked by the same host (`SRVNIX05`) and user (`root`), with a clear sequence from file extraction to execution and detection of BPFDoor. No evidence links this chain to other hosts or campaigns.

Attack Chain

Reconnaissance Resource Development **Initial Access** **Execution** **Persistence** Privilege Escalation **Defense Evasion** Credential Access

View in AI Assistant Investigate in Timeline

For Security use case, test dataset includes accurate summary of (cyber) attack

10 novel attack scenarios

- 8 Oh My Malware attack episodes (ohmymalware.com)
- 4 multi-attack scenarios (created by combining attacks in the first 2 categories)
- Benign scenarios

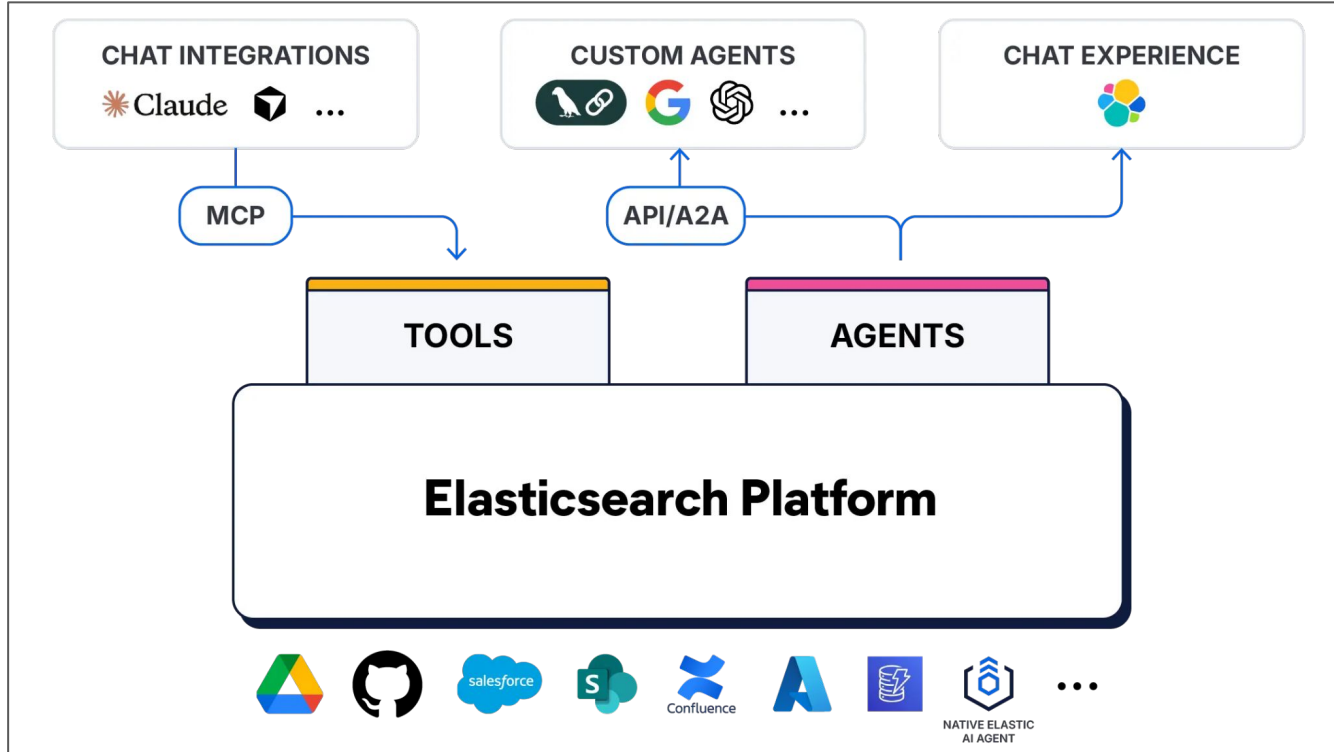
[Security] Attack Discovery

Metrics for this specific Agentic AI

Metric	Description
Precision & recall	Match alertIDs between actual and expected outputs to measure detection accuracy.
Similarity	Use BERTScore to compare the semantic similarity of the response text.
Factuality	Are key IOCs (indicators of compromise) present? Are MITRE tactics (industry taxonomy of attacks) correctly reflected?
[Domain specific] Attack chain consistency	Compare the number of discoveries to check for over- or under-reporting of the attack.

Elastic Agent Builder (Enterprise chatbot creator): Custom agents on top of existing data

Recall...



Elastic Agent Builder test datasets

Enterprise chatbot creator: Question and answer example

14 Indices using open source datasets to simulate multiple sources in KB.

- 5 Query types (analytical, text retrieval, hybrid...)
- 7 Query intent types (procedural , factual - classification, investigative; ...)

Agentic AI with RAG: Example labelled data

Enterprise data chatbot: Question and answer example

question:

'I am trying to verify my domain with Google Workspace. I need to add a TXT record to my domains DNS settings. I have confirmed that my domain is connected to Wix via pointing. What to do now?'

'To verify your domain with Google Workspace by adding a TXT record, you need to manage your DNS settings with your domain host, not Wix, since your domain is connected via pointing.

Follow these steps:

1. Log in to your domain host account where your DNS records are managed.
2. Add the TXT record provided by Google Workspace to your domain's DNS settings.
3. After adding the TXT record, return to the Google Admin Console and click 'Verify'.'

Elastic Agent Builder (Enterprise chatbot creator)

Metrics for this specific Agentic AI

Metric	Description
Precision & recall	Match documents/information retrieved by the agent to answer a user query vs the actual information or documents needed to answer the query to measure information retrieval accuracy.
Factuality	Are the key facts required to answer the user query present? Are the facts in the right order for procedural queries?
Response relevance	Does the response contain information that is peripheral or unrelated to the user query?
Response completeness	Does the response answer all parts of the user query? Does the response contain all the information present in the ground truth?
[Domain specific] ES QL validation	Is the generated ES QL syntactically correct? Is it functionally identical to the ground truth ES QL?

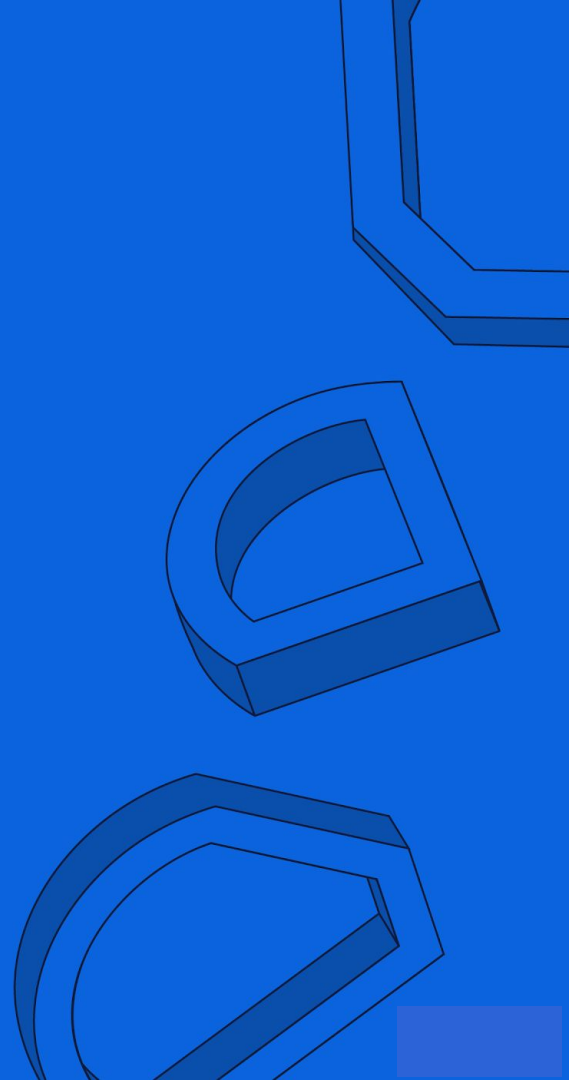
Various more agents, skills, or components each with their own evals

- Security detection rules generation AI Agent
- Security alerts RAG
- Security ES|QL detection rules generation
- Observability AI insights
- Observability AI investigations

and so on...

As agent development matures, more and more evaluations are needed

Tracing your application



Trace your application for evaluations and monitoring

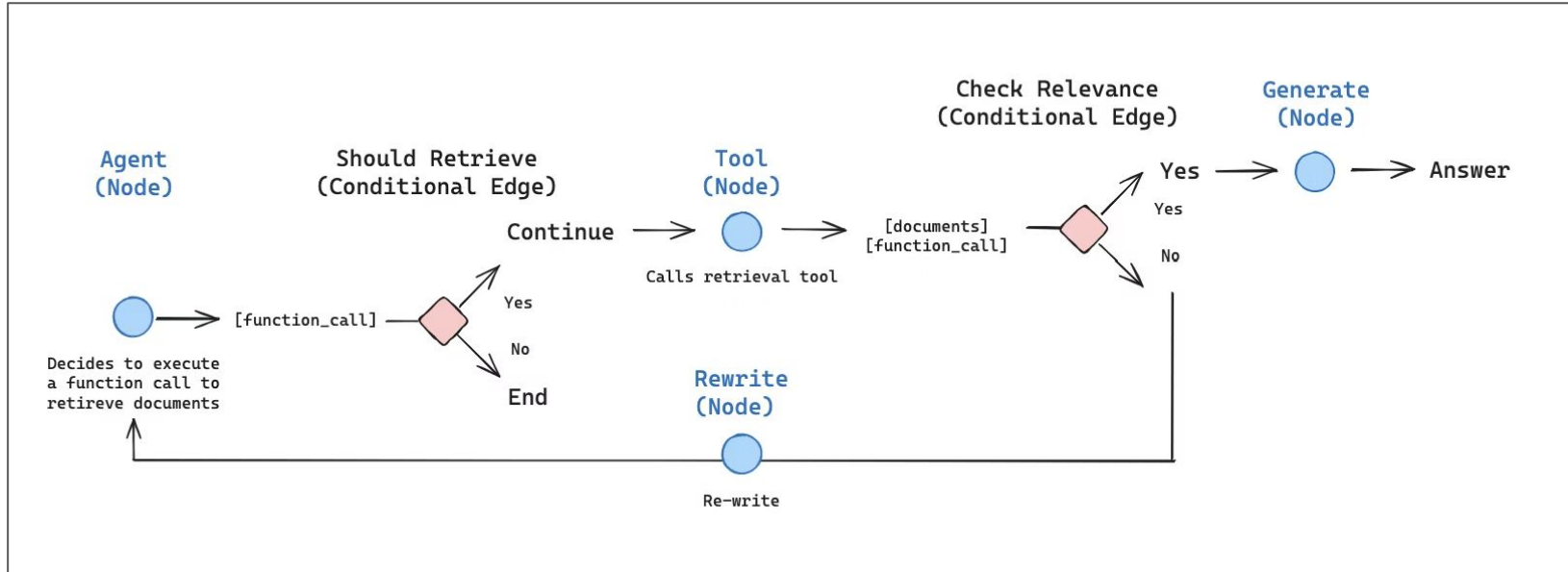
Many tools:

- LangSmith
- Phoenix
- Langfuse
- Elastic
- MLflow

Early on we tried many different tools:
As our agent development matured we started consolidating

Example: LangSmith trace (LangGraph agent)

Source: LangChain/LangGraph



Example: LangSmith trace (LangGraph agent)

The screenshot displays a LangSmith trace for a LangGraph agent. The left pane shows a tree view of the execution flow, and the right pane shows the details of the **VectorStoreRetriever** step.

TRACE

Buttons: Collapse, Stats, Filter, Show All

Tools:

- RunnableLambda (3.89s, 10,766, graph:step:4)
- RunnableLambda (3.89s, 10,766, seq:step:1)
- RunnableLambda (3.88s, 10,766)
- ESQLKnowledgeBaseTool (3.88s, esql, query-generation, knowledge-base)
- ESQLKnowledgeBaseTool (3.88s)
- VectorStoreRetriever** (0.17s, retriever, elasticsearch)
- StuffDocumentsChain (3.71s, combine_documents)
- LLMChain (3.71s, combine_documents)
- ActionsClientChatOpenAI (2024-02-15-preview, 3.71s, 10,766)
- ChannelWrite<input, steps, agentOutcome, messages, chatTitle, conversation, tools> (0.00s, seq:step:2)
- agent (3.79s, 556, graph:step:5)
- RunnableLambda

VectorStoreRetriever

Buttons: Run, Feedback, Metadata

Input

```
1 {
2   "query": "How to generate an ES|QL query to search for unusually
3     large command line strings in logs-*?"
}
```

JSON

Output

```
pack/plugins/elastic_assistant/server/knowledge_base/esql/documentatio
n/esql_rest.asciidoc"
43 }
44 },
45 {
46   "pageContent": "[discrete]\n[[esql-keep]]\n=== `KEEP`\n\nThe
`KEEP` command enables you to specify what columns are returned and
the\norder in which they are returned.\n\nTo limit the columns that
are returned, use a comma-separated list of column\nnames. The
columns are returned in the specified
order:\n\n[source,merge.styled,esql]\n----\ninclude::{esql-
specs}/docs.csv-spec[tag=keep]\n---
\n[%header.monospaced.styled,format=dsv,separator=|]\n|===\ninclude:
{esql-specs}/docs.csv-spec[tag=keep-result]\n|===\n\nRather than
specify each column by name, you can use wildcards to return
all\nncolumns with a name that matches a pattern:\n\n[source,esql]\n---
\ninclude::{esql-specs}/docs.csv-spec[tag=keepWildcard]\n---\n\nThe
asterisk wildcard (**) by itself translates to all columns that do
not\nmatch the other arguments. This query will first return all
columns with a name\nthat starts with an h, followed by all other
columns:\n\n[source,esql]\n----\ninclude::{esql-specs}/docs.csv-
spec[tag=keepDoubleWildcard]\n---\n",
47   "metadata": {
48     "source": "/Users/james/git/kibana/x-
pack/plugins/elastic_assistant/server/knowledge_base/esql/documentatio
n/esql_rest.asciidoc"
49   }
50 }
```

JSON

Example: Elastic Observability

The screenshot displays the Elastic Observability interface. On the left is a navigation sidebar with categories like Observability, Discover, Dashboards, Workflows, Alerts, and More. The main content area is divided into three sections: 'Admin and settings', 'Evaluations', and 'Converse'. The 'Converse' section is active, showing a trace for ID '9b451603ca0fb98d104466823ab9f7c1'. The trace summary indicates 99 spans, 182 hidden spans, and a total duration of 135695.5ms. A legend identifies span types: LLM (green), Tool (blue), Search (red), HTTP (orange), and Other (yellow). The trace tree shows a 'Converse' span (135695ms) containing several sub-spans: 'GenerateTitle' (3898ms), 'inference.get' (9ms), 'ChatComplete' (3881ms), 'indices.get' (2ms), 'indices.get_alias' (2ms), 'search' (4ms), 'search' (2ms), and 'ExecuteAgent' (134289ms). The 'ExecuteAgent' span further contains multiple 'inference.get' (6ms, 5ms, 3ms, 3ms, 2ms, 2ms) and 'indices.get' (<1ms) spans. On the right, a detailed view of the 'Converse' span is shown, including its duration, status (Ok), and a JSON representation of its tool output.

Admin and settings

- Alerts
- Rules
- Connectors
- Maintenance Windows
- Project performance
- Query activity **New**
- Machine Learning
- Overview
- Anomaly Detection Jobs
- Data Frame Analytics J...
- Trained Models
- Model management
- Elastic Inference

Evaluations

Runs Datasets Tracing R...

Converse

Filter traces...

Name	Feature /
Converse	-
Converse	-
Converse	-
Converse	-
Converse	-
Converse	-
Converse	-
Converse	-
Converse	-
Converse	-
Converse	-

Trace: 9b451603ca0fb98d104466823ab9f7c1

99 spans 182 hidden · 135695.5ms total

● LLM ● Tool ● Search ● HTTP ● Other

Converse 135695ms

- GenerateTitle 3898ms
 - DB inference.get 9ms
 - LLM ChatComplete 178 in / 182 out 3881ms
- DB indices.get 2ms
- DB indices.get_alias 2ms
- DB search 4ms
- DB search 2ms
- ExecuteAgent 134289ms
 - DB inference.get 6ms
 - DB inference.get 5ms
 - DB inference.get 3ms
 - DB inference.get 3ms
 - DB inference.get 2ms
 - DB inference.get 2ms
 - DB indices.get <1ms

Converse

Duration: 135695.5ms Kind: Internal

Status: Ok

Input / Output Attributes

Tool Output

```
{  "round": {    "id": "d4bd1db5-314b-41d5-bd19-e052e351a705",    "status": "completed",    "input": {      "message": "What is the most recent alert?",      "attachments": [ ]    }  }
```

Example: Phoenix

tracing

Total Traces: 431 Total Tokens: 11,033 Latency P50: 0.22s Latency P99: 7.56s

Traces	Spans	kind	name	input	output	start time	latency	total tokens	status
▼	chain	query	Does Arize support training data?	Does Arize support training data?	Yes, Arize supports training data. This data is utilized in the training environment to fine-tune th...	10/1/2023, 06:30 PM	18.32s	1040	✓
▼	retriever	retrieve	Does Arize support training data?	Does Arize support training data?	-	10/1/2023, 06:30 PM	0.83s	-	✓
	chain	reranking	Does Arize support training data?	Does Arize support training data?	-	10/1/2023, 06:30 PM	0.18s	-	✓
▼	chain	synthesize	Does Arize support training data?	Does Arize support training data?	Yes, Arize supports training data. This data is utilized in the training environment to fine-tune th...	10/1/2023, 06:30 PM	17.49s	-	✓
	llm	llm	-	-	Yes, Arize does support training data. It is used in the Training Environment where the model learns...	10/1/2023, 06:30 PM	4.28s	236	✓
	llm	llm	-	-	Yes, Arize does support training data. It is utilized in the Training Environment where the model re...	10/1/2023, 06:30 PM	3.73s	251	✓
	llm	llm	-	-	Yes, Arize supports training data. It is used in the Training Environment to fine-tune the model's p...	10/1/2023, 06:30 PM	3.51s	261	✓
	llm	llm	-	-	Yes, Arize supports training data. This data is utilized in the training environment to fine-tune th...	10/1/2023, 06:30 PM	5.95s	292	✓

Example: LangSmith trace easy "Add to Dataset" feature

The screenshot displays the LangSmith interface for a tracing project. On the left, a sidebar shows the project name '9-10-log4jvuln' and a list of runs, all named 'Attack discovery' with green checkmarks. The main area shows a detailed trace for one of these runs. The trace is titled 'Attack discovery' and shows a duration of 26.72s and 16,579 tokens. The input is 'openai'. The trace steps include: '_start_' (0.21s), 'ChannelWrite<attackDiscoveries,attac...' (0.07s), 'ChannelWrite<branch:__start__:edge:r...' (0.07s), and 'retrieve_anonymized_alerts'. On the right, a panel for 'Attack discovery' shows the 'Run' tab with a 'Feedback' dropdown and 'Metadata' options. A context menu is open over the 'Add to Dataset' button, with 'Add to Annotation Queue' also visible. The 'Rendered Output' section shows a prompt for an 'AttackDiscoveryPrompt' with a detailed instruction: 'You are a cyber security analyst tasked with analyzing security events from Elastic Security to identify and report on potential cyber attacks or progressions. Your report should focus on high-risk incidents that could severely impact the organization, rather than isolated alerts. Present your findings in a way that can be easily understood by business executives of the...'. The 'Metadata' section on the right shows: START TIME: 12/23/2024, 02:38:44 PM; END TIME: 12/23/2024, 02:39:11 PM; TIME TO FIRST TOKEN: N/A; STATUS: Success; TOTAL TOKENS: 16,579 tokens.

Shared evaluation framework: Overview



Why? As agent development matures, more and more evaluations are needed

- Cumbersome to keep building evaluations from scratch
- Different dev teams can collaborate and maintain
- Quick to spin up fully featured evals

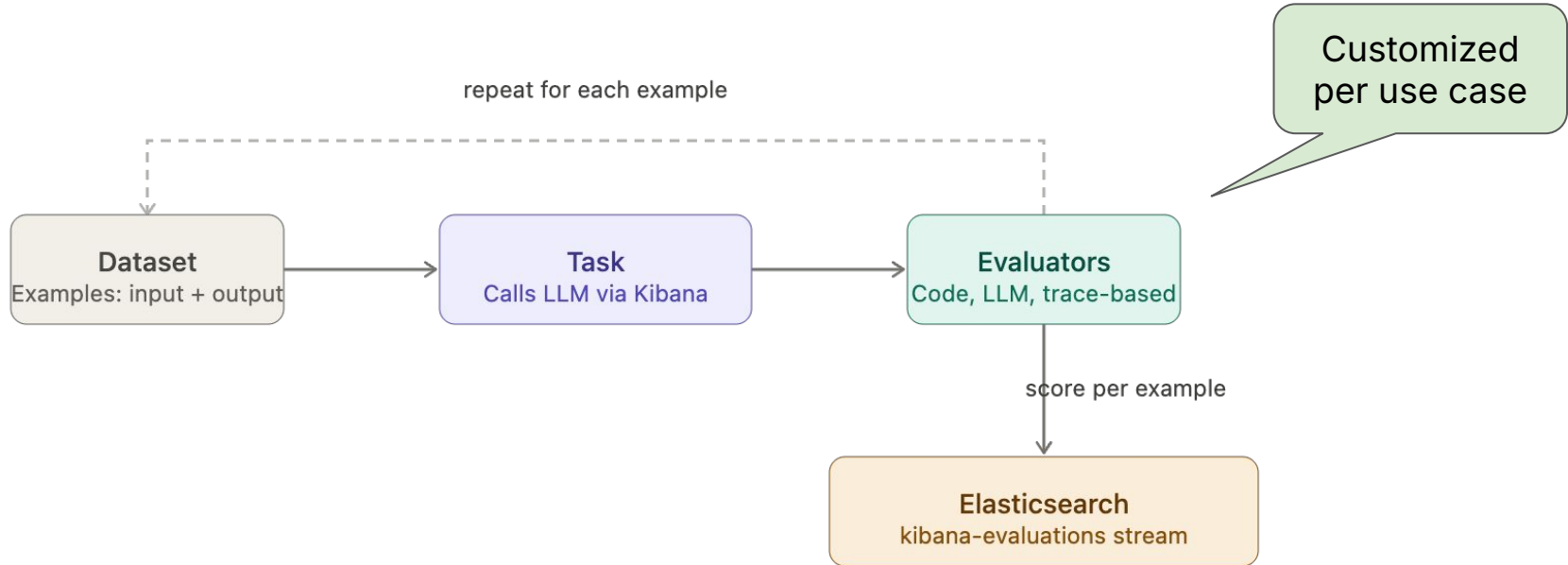
Security org and Agent Builder org previously built out completely separate eval frameworks

Shared toolkit for different Agentic AI teams

- Import datasets
- Trace based evaluators (token usage, latency, tool calls)
- RAG evaluators (Precision@K, Recall@K, F1@K)
- Custom evaluators

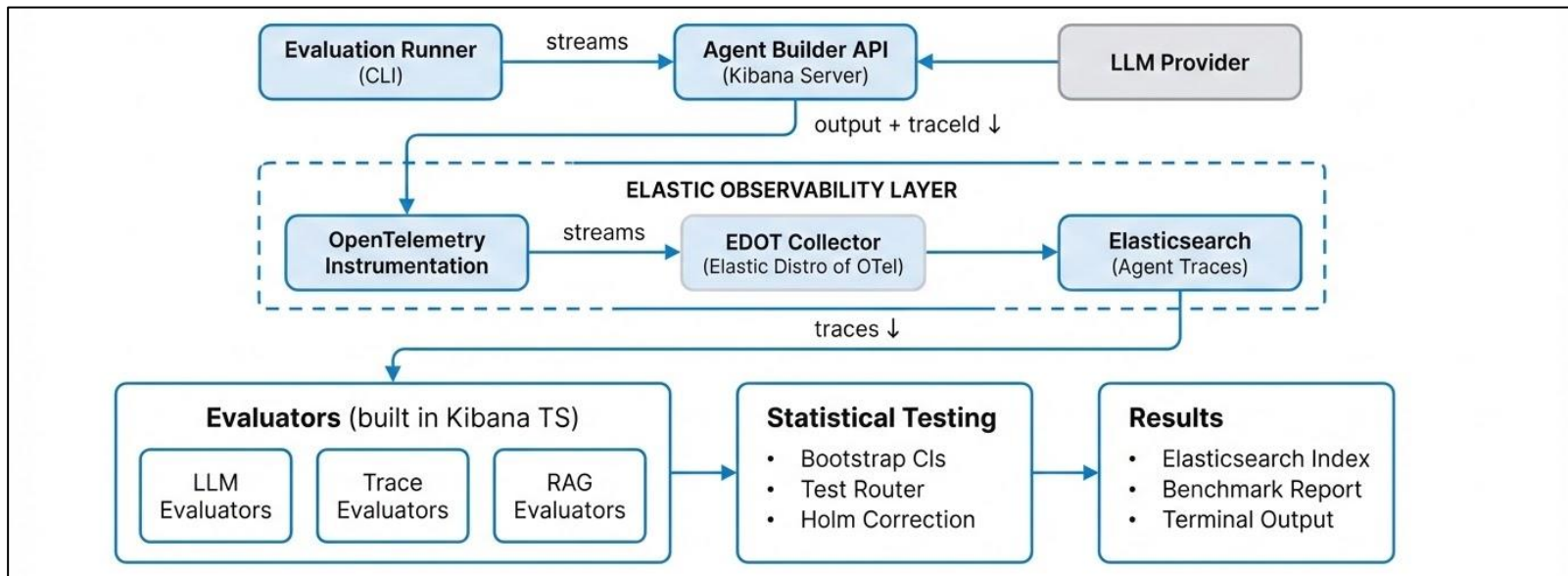
Prerequisite for evals: Tracing/ telemetry

Customizable building blocks



Orchestrated with Scout

Kibana's UI and API test framework built on top of Playwright



Elastic/Kibana Scout:

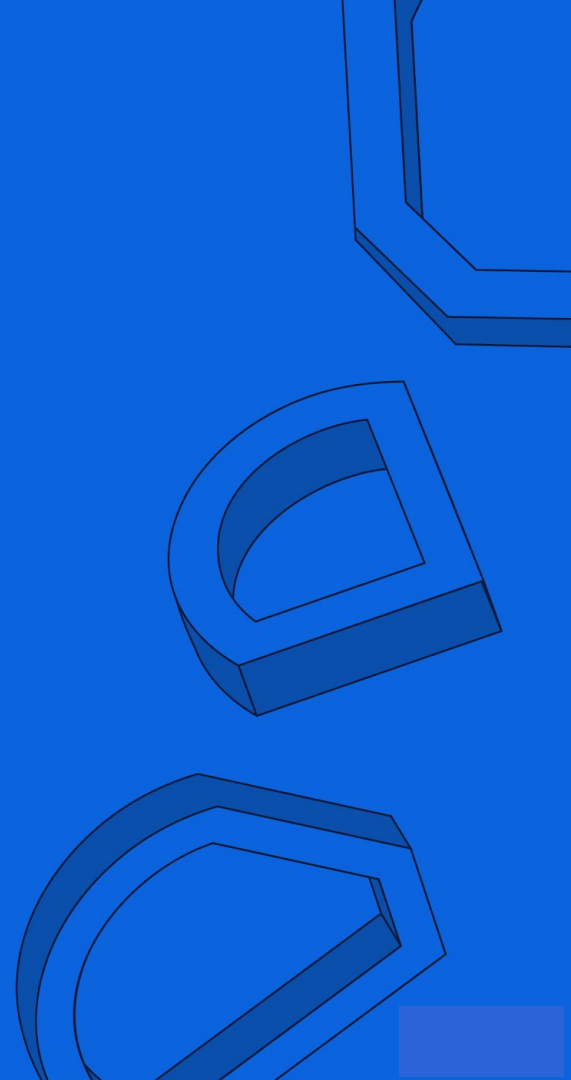
<https://www.elastic.co/docs/extend/kibana/scout>

Dev mode: outputs terminal results

```
Model: us.anthropic.claude-3-7-sonnet-20250219-v1:0 (Claude/Anthropic)
info [scout-worker]
===== EVALUATION RESULTS =====
```

Dataset	# Examples (repetitions x examples)	Factuality	Groundedness	Relevance	Sequence Accuracy
onechat: default-agent-text-retrieval-queries	3 x 29	52.1% mean: 0.521 median: 0.640 std: 0.292 min: 0.000 max: 0.923	89.8% mean: 0.898 median: 0.997 std: 0.289 min: 0.000 max: 1.000	51.8% mean: 0.518 median: 0.500 std: 0.212 min: 0.000 max: 1.000	90.1% mean: 0.901 median: 1.000 std: 0.265 min: 0.000 max: 1.000
onechat: default-agent-analytical-queries	3 x 15	42.7% mean: 0.427 median: 0.166 std: 0.443 min: 0.000 max: 1.000	97.6% mean: 0.976 median: 1.000 std: 0.149 min: 0.000 max: 1.000	64.8% mean: 0.648 median: 0.714 std: 0.272 min: 0.222 max: 1.000	100.0% mean: 1.000 median: 1.000 std: 0.000 min: 1.000 max: 1.000
onechat: default-agent-hybrid-queries	3 x 9	24.3% mean: 0.243 median: 0.000 std: 0.341 min: 0.000 max: 0.944	88.7% mean: 0.887 median: 1.000 std: 0.319 min: 0.000 max: 1.000	60.0% mean: 0.600 median: 0.600 std: 0.332 min: 0.056 max: 1.000	100.0% mean: 1.000 median: 1.000 std: 0.000 min: 1.000 max: 1.000
onechat: default-agent-unanswerable-queries	3 x 13	25.2% mean: 0.252 median: 0.224 std: 0.248 min: 0.000 max: 0.766	65.6% mean: 0.656 median: 0.992 std: 0.472 min: 0.000 max: 1.000	62.6% mean: 0.626 median: 0.500 std: 0.240 min: 0.250 max: 1.000	92.0% mean: 0.920 median: 1.000 std: 0.200 min: 0.200 max: 1.000
onechat: default-agent-ambiguous-queries	3 x 4	22.8% mean: 0.228 median: 0.148 std: 0.251 min: 0.000 max: 0.819	99.0% mean: 0.990 median: 1.000 std: 0.022 min: 0.924 max: 1.000	79.4% mean: 0.794 median: 0.938 std: 0.248 min: 0.400 max: 1.000	100.0% mean: 1.000 median: 1.000 std: 0.000 min: 1.000 max: 1.000
Overall	3 x 70	39.8%	87.4%	59.2%	94.4%

Shared evaluation framework: Building blocks



Shared evaluation building blocks

- Precision/Recall
- Semantic similarity e.g. typical answer is similar to existing docs
- Factuality e.g. not hallucinated product ID
- LLM-as-judge shared tooling

We compared commonly used metrics in ad-hoc evaluations, and started pulling them out

Factuality (customize based on application)

```
def compute_factuality_score(  
    expected_tactics: list, actual_tactics: list, iocs: list, details: str  
) -> float:  
    """  
    Compute factuality score based on matched MITRE tactics and IOC values in the response.  
  
    Args:  
        expected_tactics (list): Expected MITRE attack tactics.  
        actual_tactics (list): Tactics extracted from actual response.  
        iocs (list): List of IOCs with associated importance.  
        details (str): Markdown details from the actual response.  
  
    Returns:  
        float: Combined score indicating factual correctness of the response.  
    """  
    weights = {"high": 3, "medium": 1}  
    tactic_score = (  

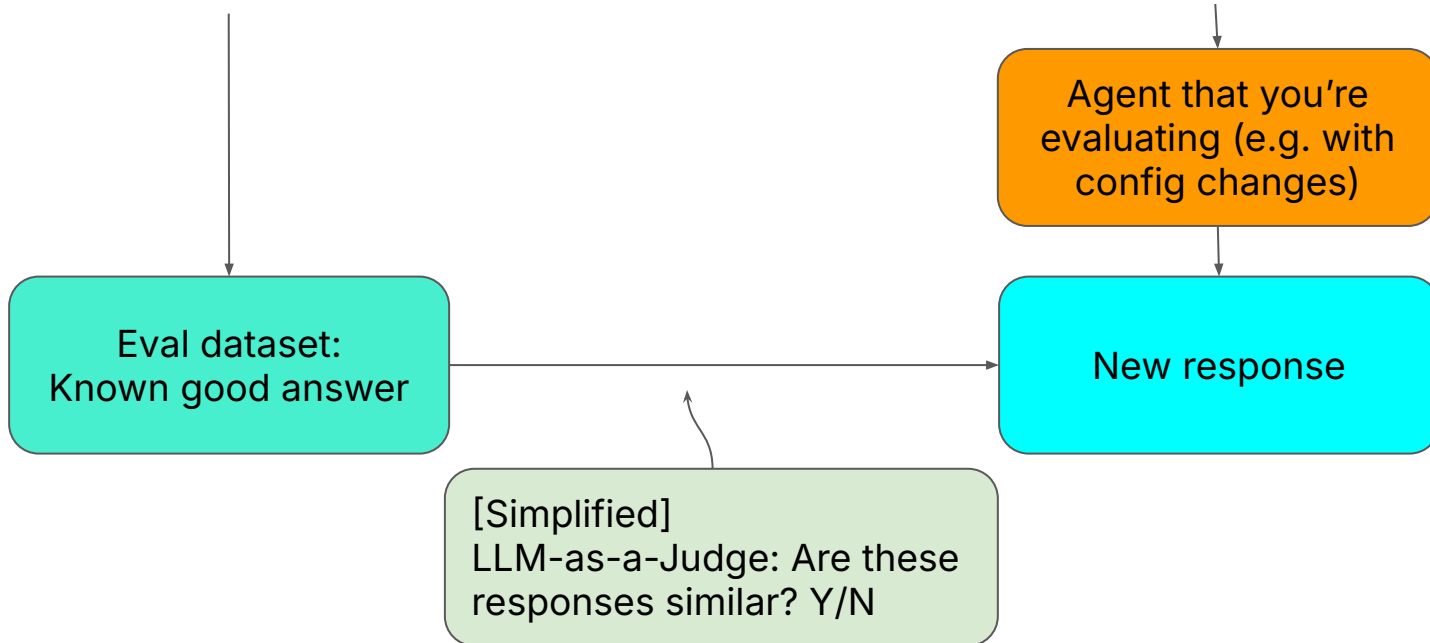
```

Early implementation for one eval suite in Python,
now since merged into shared evals in TypeScript

LLM-as-a-judge with offline evaluations on test set

Sample user question:

Generate an ES|QL query that will parse the DNS registered domain from a DNS query, count the number distinct DNS queries being made per DNS registered domain and filter for when the distinct count is greater than 5. The query should sort the results by the distinct count of queries in descending order.



LLM-as-Judge pros & cons

Using LLMs to determine if incoming data is similar to test/golden dataset

Pros:

- Scale
- Useful for tone, coherence, style, summaries, safety...
- Good for open-ended tasks
- Can judge correctness, completeness...
- Can self-explain grading using chain-of-thought

Tip: Can use structured outputs/schema,
e.g. LangSmith/LangChain [schemas](#)

LLM-as-Judge pros & cons

Using LLMs to determine if incoming data is similar to test/golden dataset

Cons:

- May not be granular enough
- Variance across runs even with the same model
- LLM might not identify crucial differences
- More guidance is needed for internal info (e.g. product IDs)
- Performance differs by models/use case
- Bias toward models from the same "family"
- Advanced math, etc.

Using the same model family may cause bias

CyberSOCEval: Benchmarking LLMs Capabilities for Malware Analysis and Threat Intelligence Reasoning (CrowdStrike, Meta)

<https://arxiv.org/pdf/2509.20166>

under test is the same, or has similarities with the set of models that were used in synthetic data generation pipelines. To ensure compliance with third party API acceptable use policies, we restricted our synthetic data generation pipelines to use a combination of Llama 3 and Llama 4 models, which could artificially boost the apparent performance of these models on the benchmarks. We attempted to mitigate this concern through a combination of complex multi-agent generation pipelines combined with extensive manual review and editing, however it remains the case that this is a potential source of bias.

Another important limitation to call out is that the contextual data provided to an AI system under test in each of

LLM-as-Judge / Model-based graders

Methods	Strengths	Weaknesses
<ul style="list-style-type: none">• Rubric-based scoring• Natural language assertions• Pairwise comparison• Reference-based evaluation• Multi-judge consensus	<ul style="list-style-type: none">• Flexible• Scalable• Captures nuance• Handles open-ended tasks• Handles freeform output	<ul style="list-style-type: none">• Non-deterministic• More expensive than code• Requires calibration with human graders for accuracy

Source: Anthropic

<https://www.anthropic.com/engineering/demystifying-evals-for-ai-agents>

Rules based evaluations

Catch crucial points you can't let LLM-as-a-Judge get wrong

Examples

- Using code (e.g. Python, TypeScript) to check if generated .json is valid
- Heuristics: If generated output doesn't have x, y, z categories for this sample, it's incorrect/correct
- Programmatic syntax checks for generated code

Example: Check ES|QL (a query language in Elastic)

Rules based evaluations: Pros and cons

Catch crucial points you can't let LLM-as-a-Judge get wrong

Pros:

- No ambiguity, catches some weaknesses of LLM-as-a-Judge
- Cheap
- Fast

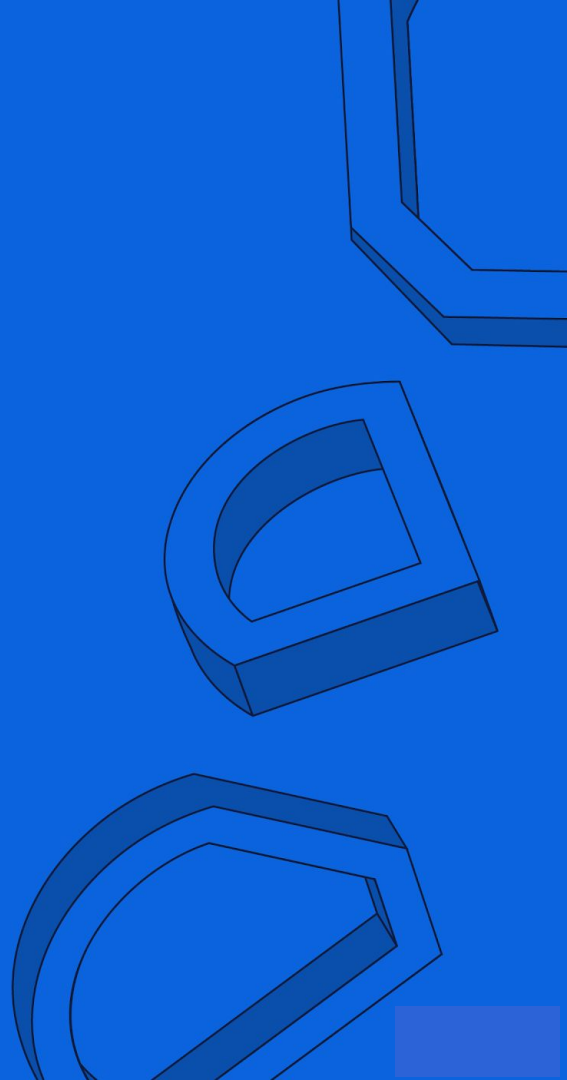
Rules based evaluations: Pros and cons

Catch crucial points you can't let LLM-as-a-Judge get wrong

Cons:

- Hard to scale for large problem space
- Can't capture nuance
- Weaker at open ended tasks

**What couldn't we
abstract?**



Can't abstract bespoke data creation

But abstracted loading of bespoke datasets

- Still requires domain expertise to review
- Product input on what are positive vs. negative examples we must pass
- Regression is different for different agents

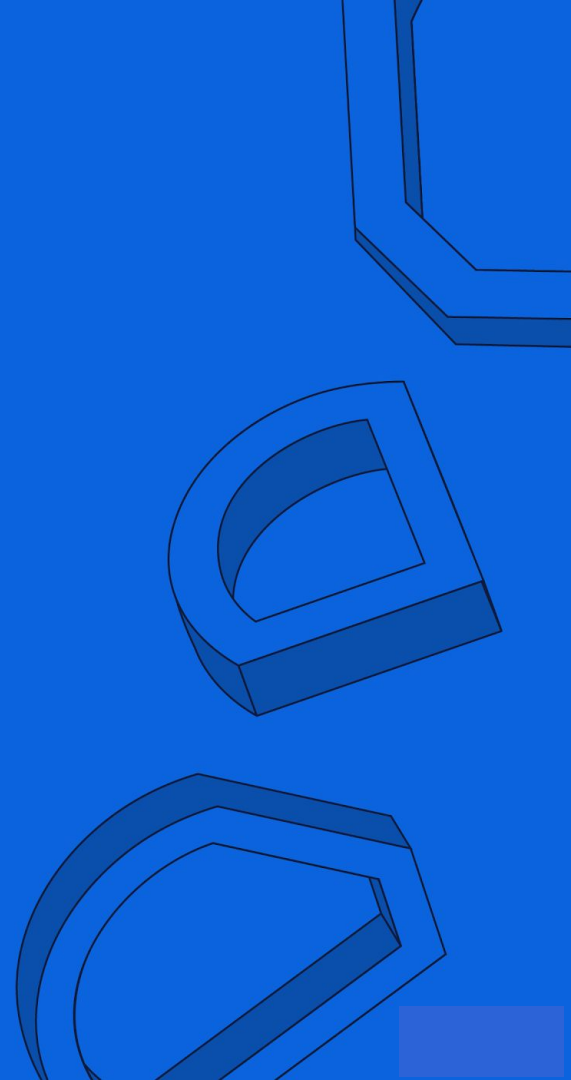
Further calibration of evaluators

"A well-calibrated evaluator should be consistent. If the baseline wins in the first evaluation, it should also win in the second. If the judgment flips, the outputs are perhaps too similar to distinguish, and we can mark these as ties rather than forcing a noisy decision." - Eugene Yan

Be mindful that your evaluators agree generally with human evaluators (or your target behavior)

We make it easy to spin up the evaluators,
calibration is still the responsibility of each team

Key takeaways



If you're building your first agent:

- Don't skip product and domain expertise in evaluations
- Try out various tracing tools, see what works
- Start small: simple dataset creation, ad-hoc evals is fine

If your company is still early in its agent development journey, don't over-invest in abstractions yet.
Optimize for learning and experimentation first.

If you have a few agents in production:

- If you're not making evaluations, start building at least one
- Thinking about the common metrics you're observing
- You'll eventually run into the problems of too many evals, how can you plan ahead to avoid that

Cross-team communication is key,
as agent dev in your organization matures.

Lessons we've learned

If we could do over...

- Build first, consolidate later, but would have planned for consolidation earlier
- Make it easier for feedback to improve the product
- Would have jumped into the language of the main stack earlier

If you already have your own evaluation "hub", what would you do over?

We share eval results for select use cases publicly

Proprietary models

Models from third-party LLM providers.

Model	Alerts	Security Knowledge	ES QL Query Generation	Knowledge Base Retrieval	Attack Discovery
Opus 4.6	8.9	9.5	8.5	8.42	8.7
Sonnet 4.5	8.6	7.6	7.7	7.23	8
Opus 4.5	9	8.2	7.5	7.94	8.5
GPT 5.2	8.6	6.6	8	6	8.5
Sonnet 4	7.5	7.4	8	7.85	7
Sonnet 4.6	9.3	9.5	8.4	7.45	Not recommended
Sonnet 3.7	7.4	6.9	6.1	7.04	7
GPT 5.1	9.3	4.3	7.2	6	6.5

Get

- ✓ 14-day
- ✓ All featu
- ✓ No setu

On this page

- Proprietary r
- Open-sourc

Source:

<https://www.elastic.co/docs/solutions/security/ai/large-language-model-performance-matrix>

Shoutouts

Elastic team members that made these features possible

- Abhimanyu Anand
- Srdjan Lulic
- Kirti Sodhi
- Gus Carlock
- Garrett Spong
- Andrew Macri
- and more

Thank you!
Q&A

Appendix

> export



JUNE 1-2, 2026

BOSTON UNIVERSITY, GEORGE SHERMAN UNION
BOSTON, MA

< SESSION >

Building Reusable Evaluation Frameworks for Agentic AI Products



Susan Chang

Principal Data Scientist
@Elastic