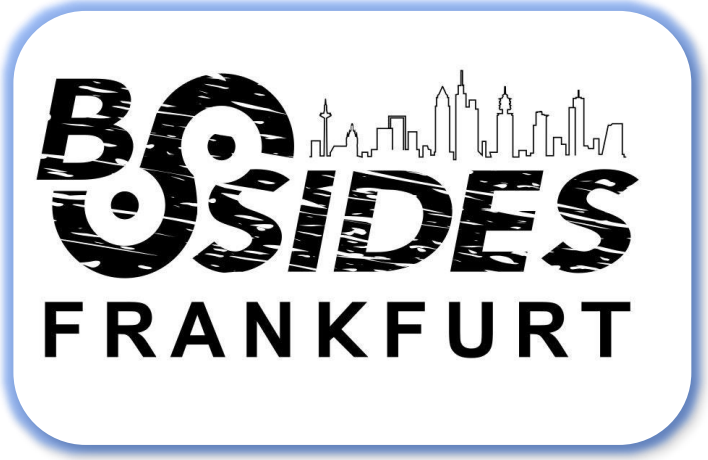


Lessons from building an

AI Powered SOC

The Good, The Bad, and the Scary lessons learned

Aaron Jewitt · Detection Engineering & Security Operations, Elastic

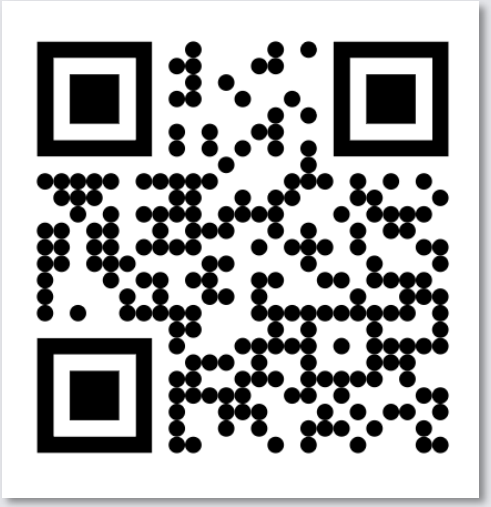
BSides Frankfurt · 10 September 2026

Whoami

Aaron Jewitt

Principal Detection Engineer · Elastic InfoSec

- 20+ years of hands on offensive and defensive security experience
- Build and run the detections that defend Elastic's own network
- I've written lots of blogs about detection engineering at Elastic
  - https://www.elastic.co/security-labs/author/aaron-jewitt



linkedin.com/in/aaron-jewitt

Definitions

Making sure we are all on the same page

WHAT WE WILL DEFINE

- Tokens
- Context window
- Models
- Tools & MCP
- Skills

What is a Large Language Model (LLM)?

A model is a fixed set of weights that predicts the next token. It has no memory between calls, no access to your data, and no ability to act. Everything else is scaffolding you build around it.

FROZEN AT TRAINING

Knowledge stops at the training cut-off. Anything newer, or internal, must arrive in the prompt.

STATELESS BY DEFAULT

Each call starts blank. History is re-sent every time, so context is an engineering problem.

PROBABILISTIC, NOT LOOKUP

It generates plausible text, it does not retrieve facts. Confidently wrong looks the same.

Not all models are created equal

Rule of thumb: Use the cheapest model that gets the job done accurately

Pro Tip: Save money by breaking a big task into smaller parts to use smaller models

FRONTIER

Deepest reasoning, largest context, highest cost. They have the largest knowledge base, ok for making judgement calls.

SMALL AND FAST

Much faster and cheaper, but not nearly as smart. These are good for quick answers on a small prompt.

SPECIALIZED

Some models are specialized to be very good at one thing. Extracting data from images and videos, converting html to markdown, etc.

Model size represents the number of parameters per Billion.

The model must be loaded in high speed RAM, this is the hardware limiter of model size

The Model is a Goldfish!

- Every call to an LLM is atomic
- The model has no memory of its previous conversations
- This means that the harness has to simulate memory by adding context from previous requests to each LLM call

What is a token?

> Models do not read words. Text is chopped into tokens, fragments of a few characters. Every limit you hit and every euro you spend is counted in tokens.

CHUNKS OF TEXT

Roughly 4-7 characters, including whitespace. Longer German words require more tokens. Log lines, base64 and raw JSON split far less efficiently than markdown.

THE CONTEXT WINDOW

The total tokens a model can hold at once: your prompt, the tool output and its answer. Maximum context window size is determined by model size and the amount of available High Bandwidth Memory

WHAT YOU PAY FOR

Billing is per tokens in and out. Sending unnecessary data to the model has a very real impact to cost.

Models prefer plain text and markdown

```
This is an example of tokenizing
```

7 tokens

```
<p>
    This is an example of tokenizing
</p>
```

14 tokens

Lesson Learned: Don’t send raw json or xml logs to the model. Save money by parsing it first.

What is an Agent?

> An agent is a program that uses an LLM model that picks its own next step. You give it a goal and a set of tools and skills, then it loops until the goal is met or you stop it. Each agent has its own preconfigured instructions.

GOAL

You state the outcome, not the path. “Triage this alert”, not “run these twelve steps in this order.”

TOOLBOX

A bounded set of actions it may call: queries, enrichment, case updates. The toolbox is your control surface.

LOOP · OODA

Observe, orient, decide, act, then do it again until the goal is met, a budget is spent, or it hands off to a human.

What is a tool?

> A tool is a function you expose to the model, described in plain language. The model uses tools to take actions such as API requests or running local programs

A DESCRIBED FUNCTION

Name, purpose and typed parameters. The description is prompt text, so vagueness misfires.

MCP, API, SCRIPTS, ETC.

These are ways for your model to take actions on the world around it.

YOUR CONTROL SURFACE

Hooks can control what tools are allowed to do. Limits on models are just suggestions.

What is a skill?

> Additional prompt instructions and tools that are loaded on-demand by an agent. A skill looks a lot like an agent, but it doesn't run on its own.

PROCEDURE, NOT PLUMBING

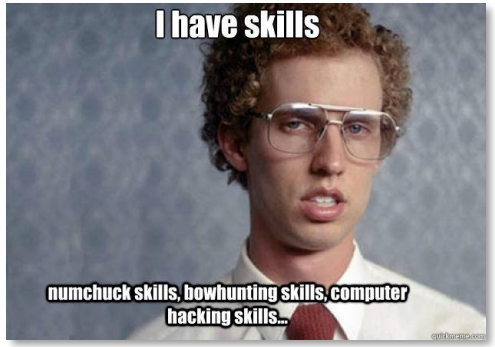
Your triage runbook as text: the steps, the thresholds, the phrasing for the case note.

LOADED ON DEMAND

Skills stay out of context until relevant. The agent has a smaller prompt at run-time

REUSABLE BY AGENTS

Create a skill once and share it among multiple agents.



Workflows, scripts, and harnesses

Control your agents and LLMs using old fashioned deterministic code

YOU OWN THE PATH

The sequence of steps lives in code you reviewed, not in a model's next-token guess. Same input, same route, every time.

LLM AS A STEP

The model classifies, summarises and drafts. It never picks the tools, the order, or when to stop.

TESTABLE AND CHEAP

Deterministic paths can be unit tested, replayed and audited. Token spend and latency stop being a surprise.

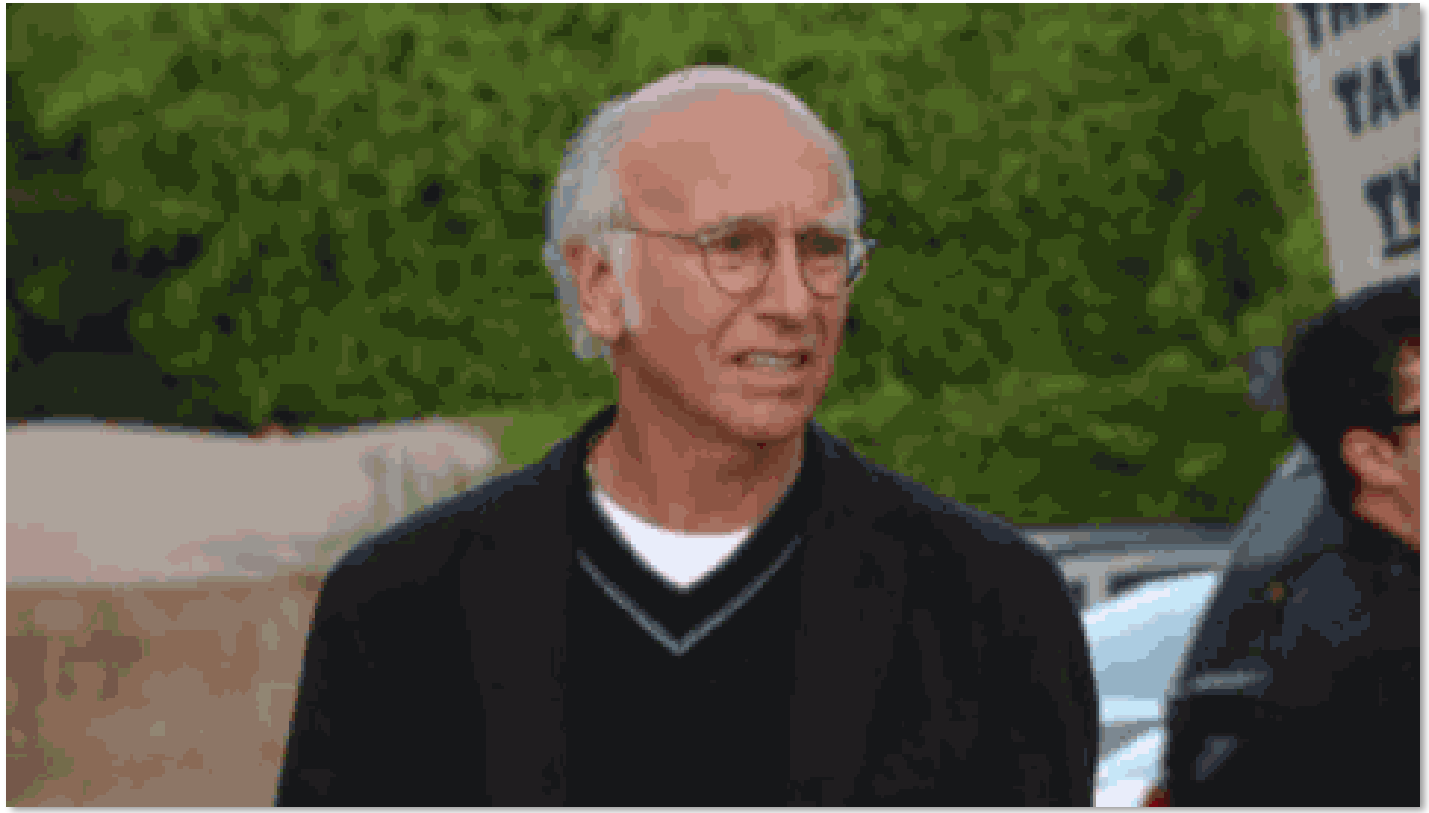
Agents vs Workflows

WORKFLOW / SCRIPT		AGENT
You write it, at build time	CONTROL FLOW	The model decides it, at run time
Fully known in advance	THE PATH	Discovered as it runs
Always the same path	SAME INPUT	Maybe a different path
The last step finishes	STOPS WHEN	The goal is met, or you stop it
Data in the exact expected shape	YOU SUPPLY	A goal and a toolbox
Optional: just one step	THE LLM	Always: it is the driver

In a workflow you control the decisions. In an agent you give it the initial instructions that it may or may not follow.

*My AI SOC Maturity model

** I made this up for this talk because I couldn't find anything better.*



Four levels of AI maturity in a SOC

0 **Shadow AI**
Unmanaged

No policy, or a policy with no sanctioned tools. Personal accounts are used.

1 **Assisted**
Manual

Approved AI powered assistant, up to and including full tool and data access.
Triggered through Manual interactions from the users

2 **Automated**
Integrated

AI running unprompted inside the pipeline you already had. LLMs are used without human interaction

3 **Optimized**
Restructured

The SOC is built around AI capabilities. Your detections, response, thread intel, vulnerability management systems and processes are designed for AI.

Level 0: Shadow AI

- Your employees are using AI. If you aren’t providing it, they are using their own personal accounts.
  - With no enterprise provided tools this is causing sensitive data leakage and potential legal headaches
- If you tell them not to use personal AI, but you don’t actively monitor and block Shadow AI they are still doing it.
  - Completely blocking Shadow AI is really hard to do

Four levels of AI maturity in a SOC

0 — Shadow AI
Unmanaged

No policy, or a policy with no sanctioned tools. Personal accounts are used.

1 — Assisted
Manual

Approved AI powered assistant, up to and including full tool and data access.
Triggered through Manual interactions from the users

2 — Automated
Integrated

AI running unprompted inside the pipeline you already had. LLMs are used without human interaction

3 — Optimized
Restructured

The SOC is built around AI capabilities. Your detections, response, thread intel, vulnerability management systems and processes are designed for AI.

Level 1: Assisted – The AI powered Chat Bots

A human starts every interaction, decides what to ask, and judges every answer.

HUMAN 01

Ask

Analyst frames the question and pastes in the context

AI 02

Generate

Model returns a summary, a query, or an explanation

HUMAN 03

Judge

Analyst validates every answer before trusting it

HUMAN 04

Act

Analyst runs the containment or closes the case

The Humans become faster, but they are still the bottleneck

The assistant helps the human

A better bot can do more to help, but it doesn’t work on its own

- Create queries and gather data
- Follow playbooks for response
- Summarize cases and create Timelines

A generic bot is confidently wrong

A better model only improves the grammar, not the accuracy

HUMAN ANALYST

Says ‘I don’t know’ when the data isn’t there

GENERIC BOT

Confidently gives the wrong answer with no evidence

The bot needs the same data and context a human needs

Not all Assistants are created equal

Capability compounds as each level inherits everything below it.

		CAPABILITY STACK			
		Custom prompt	Tools	Skills	Context engine
1	**Copilot / Gemini / Claude / ChatGPT handoff** no custom context, tools, or memory. Confidently wrong.	○	○	○	○
2	**Prompted assistant** Custom prompts shape reasoning; no tools or memory.	✓	○	○	○
3	**Retrieval Augmented Generation (RAG) assistant** Uses internal live data to answer questions.	✓	✓	○	○
4	**Skilled assistant** Loadable skills act: close alerts, build dashboards.	✓	✓	✓	○
5	**Learning assistant** Context engine learns from human feedback over time.	✓	✓	✓	✓

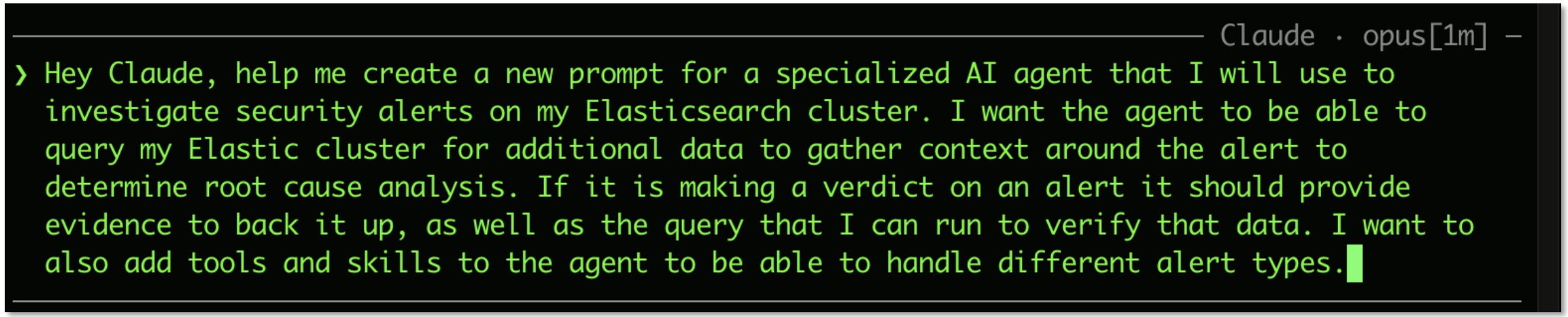
Five things a good Assistant needs

Building a good AI chatbot isn't as hard as it sounds

- A Harness to run everything
- Specialized prompt with your instructions
- Tools that give it access to live data
- Skills loaded on demand to give it flexibility
- A place to store and grow its knowledge base

Creating the prompt is easy

Make the AI build the AI



Your first prompt will be wrong

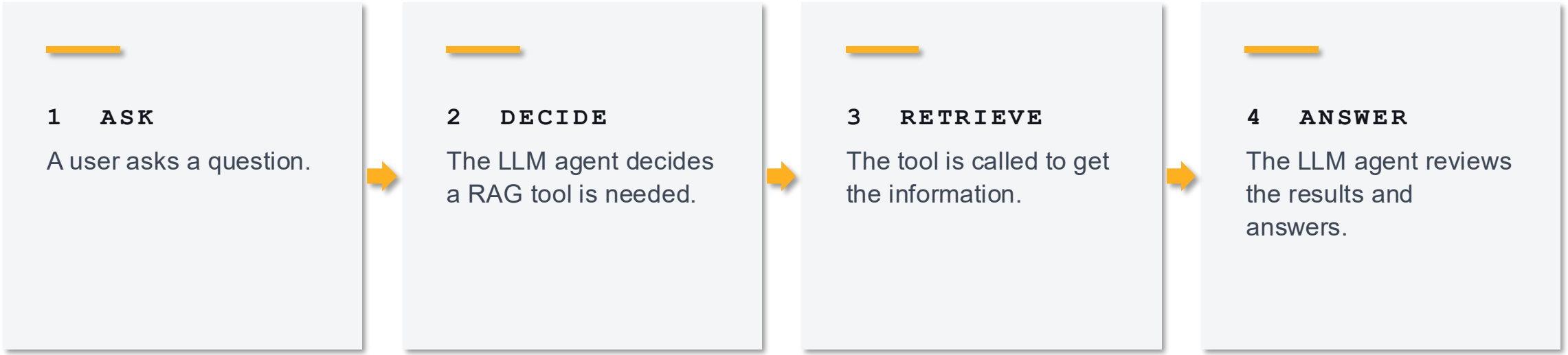
Build, test, improve, repeat

- Use AI to draft it from your requirements
- Test it on use-cases where you know the answer
- Use the bad answers to feedback to the prompt
- You will never be 100% done

RAG: stop the model from guessing

Retrieval Augmented Generation is often just an API call

A model only has knowledge about its training. RAG gives the agent a way to query live data to answer from facts instead of guesses.



Give your Assistant some Skills

Skills give you flexibility without overloading your default prompt

- Loaded only when needed
- Build a skill per data source
- Update your skills regularly

Option 1: Use a commercial Harness

Buy it

- Use a LLM Harness: Claude, Codex, Cursor, OpenCode
  - Or use the Agent built into your Elastic SIEM
- Connect to your SIEM and other data via MCP or API
- Give it access to context via RAG and knowledge base
- Custom skills and prompts
- Easy to build, high quality, sometimes high price.

Option 2: Build your own

Build it

- Vibecode and host your own agents using an opensource framework
- You have full control, you can use a local model, cloud hosted, or a combination
- More flexibility and probably cheaper, but more maintenance overhead

How does this help (if done right)?

Its not free, you need to justify its cost to the boss

- Threats are using AI to accelerate, try to keep up
- Bulk managing cases, alerts, and rules is easier
- It increases the Analyst performance and efficiency at every level


Disney+

What could go wrong?

False information being reported as true

This risk increases with agent quality

The better the agent, the more trust analysts give it. The more they trust it, the less they check its work.

You own what you hand over

Copy an agent report and pass it off as your own, and it becomes your report: you will be blamed for any mistakes.

What could go wrong?

With great power comes great responsibility

Closing alerts

Do you allow your agent to close alerts without human review?

What if your agent closes ALL alerts?

Response actions

Do you allow your agent to isolate hosts or remediate malware?

What if it isolates the wrong host?

What could go wrong?

Restrictions in a prompt are just suggestions

Agents work around restrictions

”Do not delete this” is a suggestion that can be ignored

Controls in the harness can often be bypassed

Restrict agents using external code

Restrict the API key permissions

Use external hooks with Human in the Loop requirements to continue

What could go wrong?

Bad assistants are a huge waste of money

People stop using a chatbot they cannot trust

Would you trust an analyst that is wrong 40% of the time? What about 20%? Without trust nobody will use the expensive AI.

Four levels of AI maturity in a SOC

0 — Shadow AI
Unmanaged

No policy, or a policy with no sanctioned tools. Personal accounts are used.

1 — Assisted
Manual

Approved AI powered assistant, up to and including full tool and data access. Triggered through Manual interactions from the users

2 — Automated
Integrated

AI running unprompted inside the pipeline you already had. LLMs are used without human interaction

3 — Optimized
Restructured

The SOC is built around AI capabilities. Your detections, response, thread intel, vulnerability management systems and processes are designed for AI.

Automate: Forget everything I just told you

- Skills are bad, don’t use them
- Agents shouldn’t use RAG
- Every LLM call costs money.
- Using an LLM should make you feel bad.

Level 2: Automated

A great chatbot does not always make a great automation.

Level 1: Chat

- Every conversation input and flow is different
- Doesn’t do anything without a human
- Needs skills available just in case
- If the output is wrong you can see it
- If your chat burns 40k extra tokens nobody notices

Level 2: Automated

- The input and output is controlled by scripts
- Skills can be built into the prompt to save tokens
- Mistakes and inefficiencies are compounded
- 40k extra tokens on every run will be noticed when the bill is due

Automation compounds: mistakes and inefficiencies happen every time

Story Time: an expensive mistake

- We built a chat agent with skills and tools to help investigate alerts
- We connected it to our SOAR: every triaged alert called the agent, and the output was written to the case.
- Quality was good. Then the bill arrived:

  +$20k in AI costs in a single month!

- **Lesson:** ”Good at chat" is not the same as "worth running on every alert".

Where are the biggest costs?

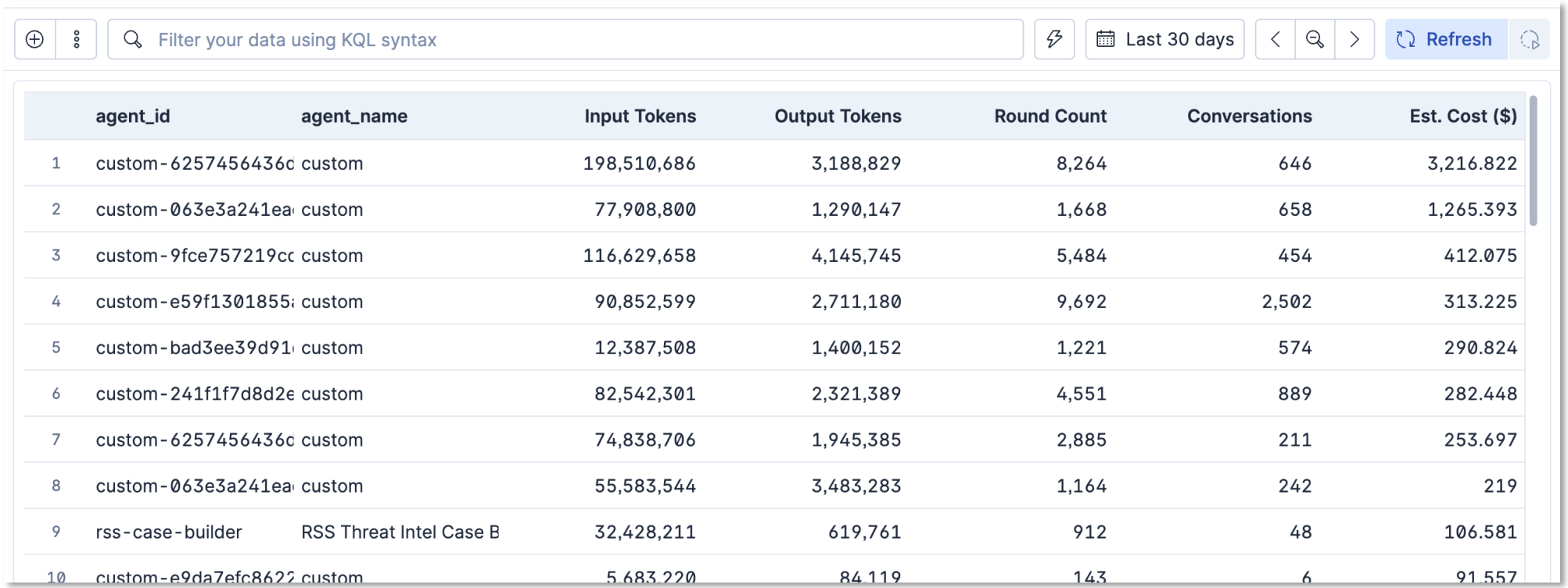
It helps when we have telemetry on all Agent activity inside of Elastic

- The #1 predictor of cost is the number of LLM calls your agent makes
- You get billed per input and output token
- Prompt size impacts token usage, but not as much as the number of round trips to the LLM

What does this cost us to run now?

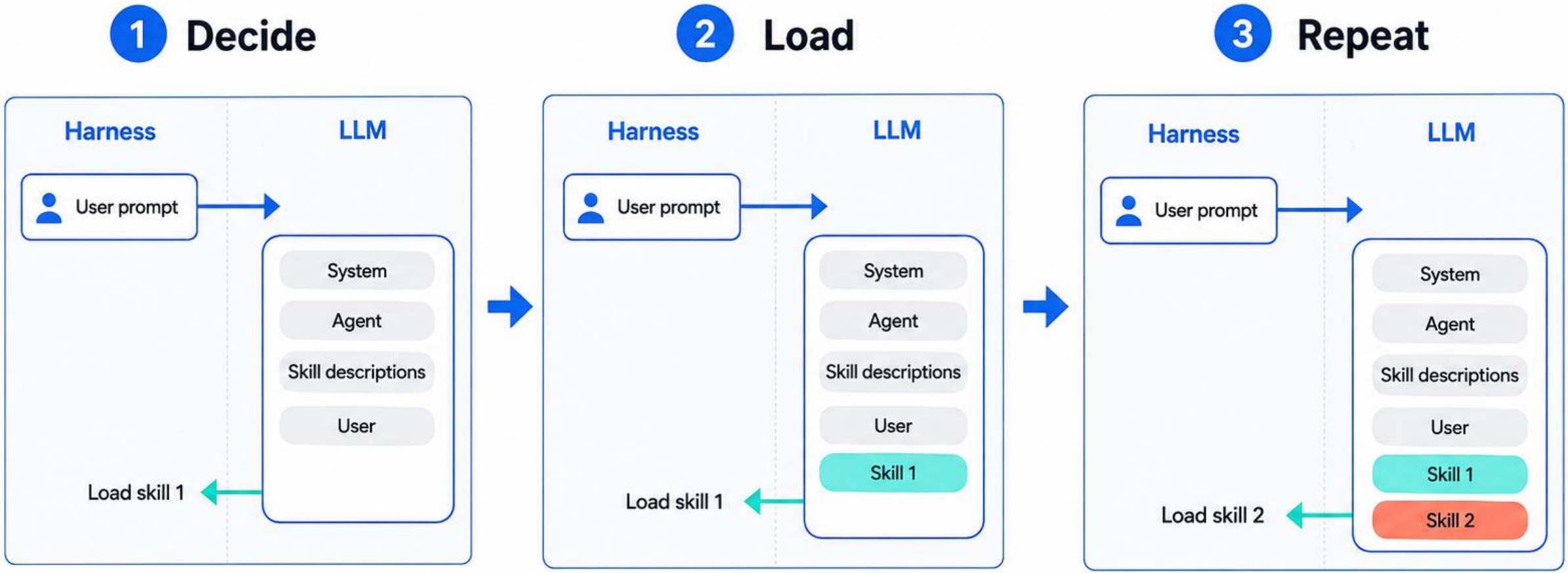
Elastic provides otel telemetry traces for all Agent activity

Our total Agentic AI spend last month was around $5500



	agent_id	agent_name	Input Tokens	Output Tokens	Round Count	Conversations	Est. Cost ($)
1	custom-6257456436c	custom	198,510,686	3,188,829	8,264	646	3,216.822
2	custom-063e3a241ea	custom	77,908,800	1,290,147	1,668	658	1,265.393
3	custom-9fce757219cc	custom	116,629,658	4,145,745	5,484	454	412.075
4	custom-e59f1301855	custom	90,852,599	2,711,180	9,692	2,502	313.225
5	custom-bad3ee39d91	custom	12,387,508	1,400,152	1,221	574	290.824
6	custom-241f1f7d8d2e	custom	82,542,301	2,321,389	4,551	889	282.448
7	custom-6257456436c	custom	74,838,706	1,945,385	2,885	211	253.697
8	custom-063e3a241ea	custom	55,583,544	3,483,283	1,164	242	219
9	rss-case-builder	RSS Threat Intel Case B	32,428,211	619,761	912	48	106.581
10	custom-e9da7efc8622	custom	5,683,220	84,119	143	6	91.557

More skills loaded, more context carried



Every LLM call carries the accumulated context.

Don’t use LLMs… until you have to

LLMs are the slowest and most expensive tool in the box.

- **Deterministic code first.** If you can do it with code, do it with code. It is free, fast and gives you the same answer every time.
- **Wrap each agent use in a preprocessing script.** Use a workflow to gather the context for your agent instead of asking the agent to do it. You agents shouldn’t need tools.
- **Agents are the last resort.** Reach for an agent only when the work genuinely needs judgement. Everything else is cheaper somewhere else.

When should you NOT use an Agent or LLM?

Don’t be lazy

Don’t use AI for anything that can be done using code. Just because the AI can answer the question doesn't mean it should.

Examples of BAD use cases for AI in automation

“Look for any alerts in the last 7 days with this same source.ip”

“Convert this string to lower case”

“extract these fields from this json”

Every LLM call costs money, so spend tokens only when you need to.

When should you use an Agent or LLM?

Judgement

Making decisions based on instructions and information in a block of text.

Structure from Chaos

Take unstructured input and structure it based on instructions. Great for reporting or summarizing

Language work at volume

Translation, determining user intent, enrichment write ups, case notes and analyst summaries.

Use the model for the step that needs judgement. Keep the workflow around it deterministic.

Split the tasks so you can pick the best model

Right-size the model to the job, not the job to the model.

One-shot needs the best

If you try to one-shot a big task you have to use the strongest model available, or the output will be poor.

Split it into pieces

Small, well-scoped steps let you pick the cheapest model that still gets the job done.

Many small calls win

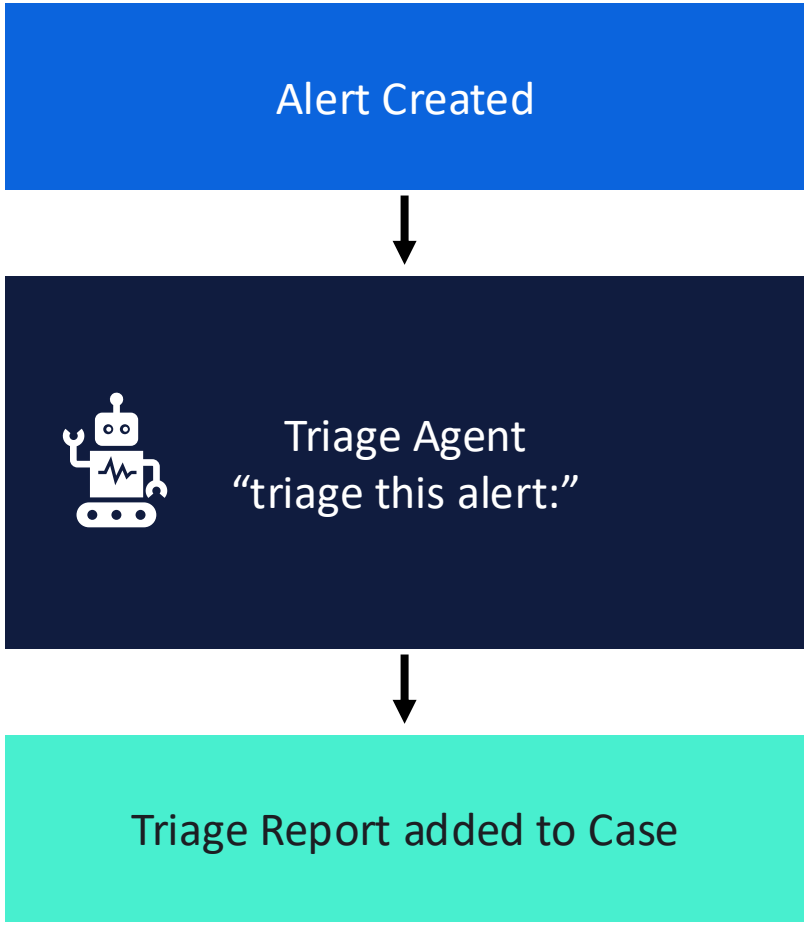
Multiple requests to smaller models are almost always cheaper than one frontier-model call.

Chat agent plugged into automation

Alert arrives
Detection rule fires.

Agent analysis
One custom prompt per alert carries the rule, raw fields and all enrichment to the agent, which returns a TP or FP verdict with rationale.

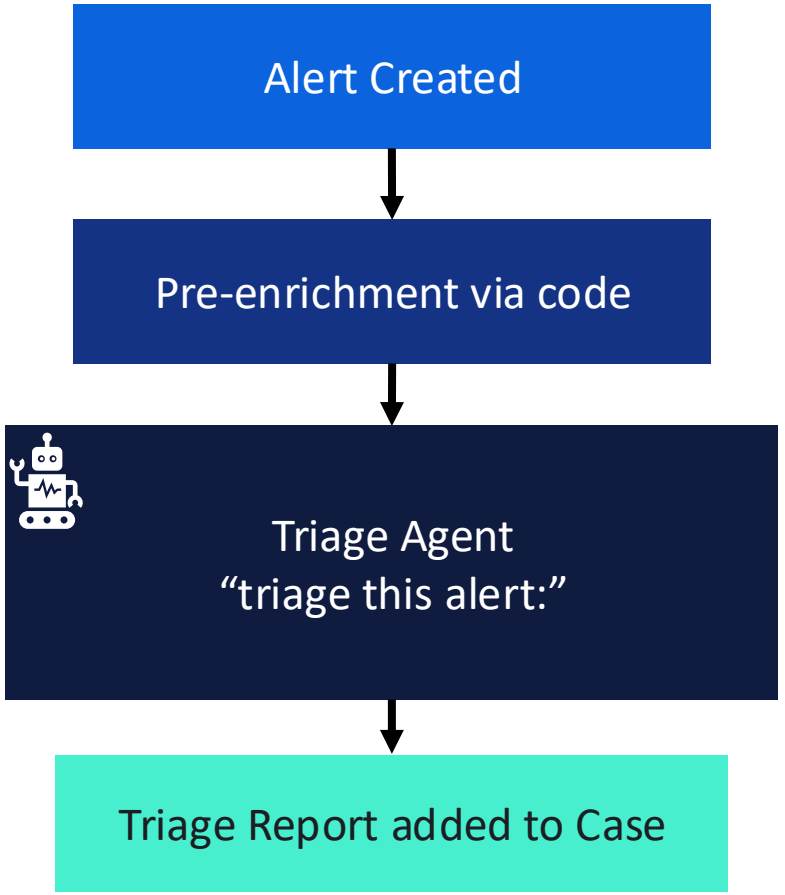
Case is created
The workflow creates a new case and attaches the report to the case for the analysts.



Better Alert triage as an agentic workflow

Deterministic enrichment
Traditional SOAR automation: fixed functions pull user, host, reputation and history context using values from the alert.

Agent analysis
Now instead of relying on tools the agent has most of the information it needs to create a report.



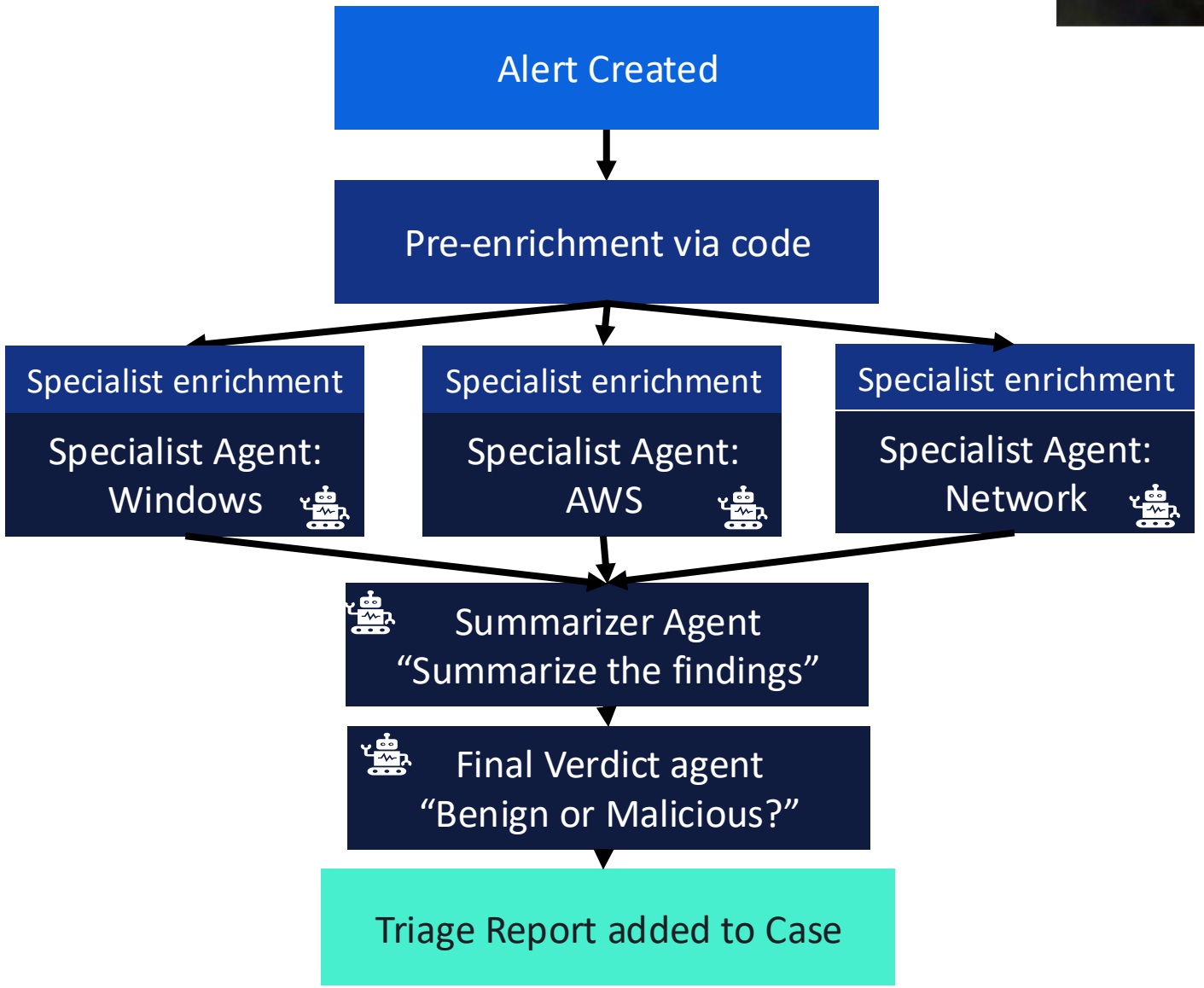
Even Better triage – Multi-agents

Multi Agent analysis
Split your analysis into multiple highly specialized agents, each with their own pre-enrichment.

Each of these agents has detailed instructions for their own type of data.

Summarizer agent
A summarizing agent combines it all together into a report. This can be a cheaper model

Report agent
Review the summary and give a verdict of True Positive or False Positive, impact of Benign or Malicious, and a confidence rating of the agent about the verdict.



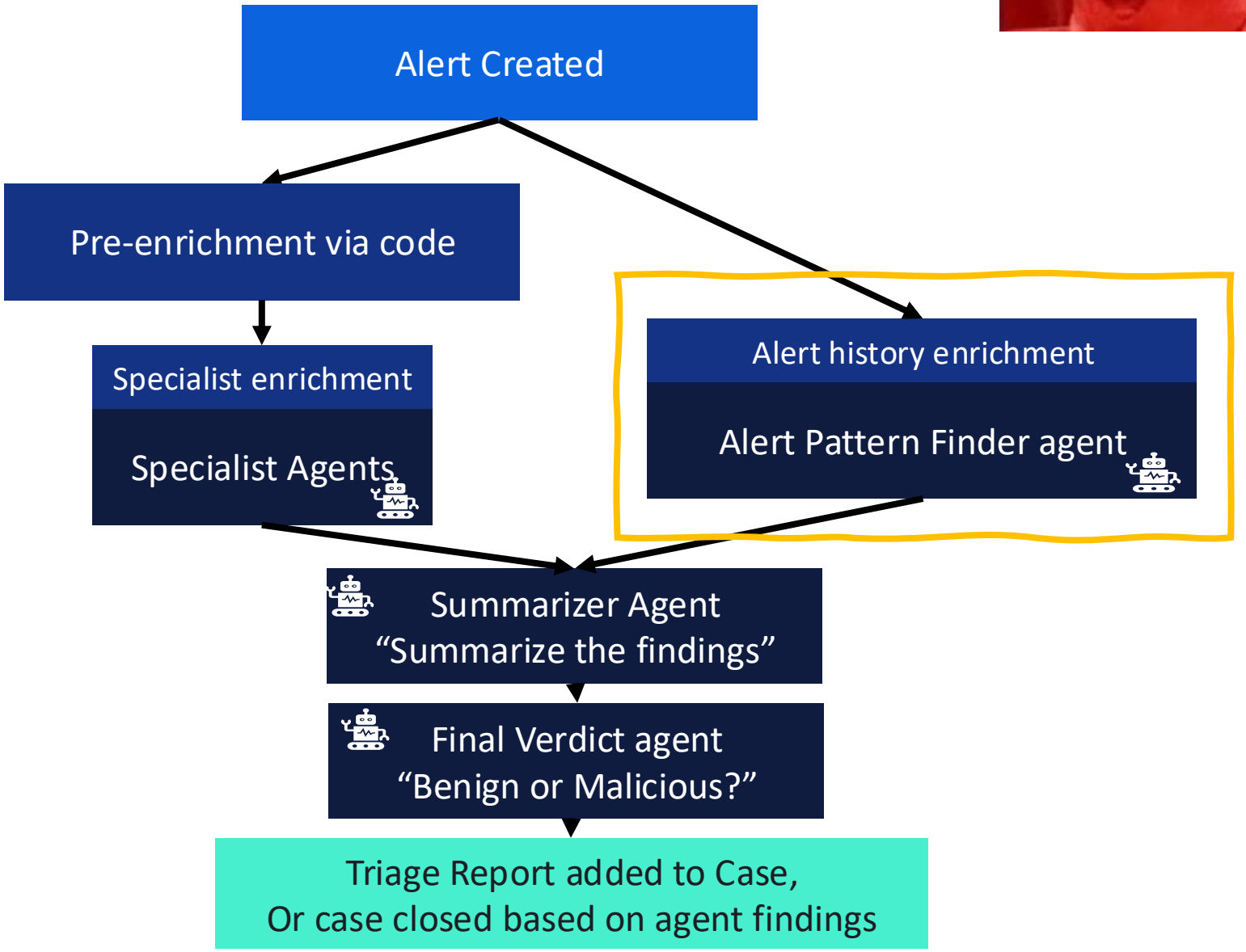
Way Better triage – self learning agents

Look at the verdict and accuracy of past assessments

In addition to the triage agents, you now have an agent that searches for any prior case related to this activity to see how that case was handled.

Extract the alert information, the investigation guide, user and host details, related cases and all comments from those cases and pass it all to the Pattern finder agent that looks for and creates a summary of the past activity.

Combine this with a way to collect feedback from humans on each case and your agent accuracy will skyrocket



Four levels of AI maturity in a SOC

0 — Shadow AI
Unmanaged

No policy, or a policy with no sanctioned tools. Personal accounts are used.

1 — Assisted
Manual

Approved AI powered assistant, up to and including full tool and data access. Triggered through Manual interactions from the users

2 — Automated
Integrated

AI running unprompted inside the pipeline you already had. LLMs are used without human interaction

3 — Optimized
Restructured

The SOC is built around AI capabilities. Your detections, response, thread intel, vulnerability management systems and processes are designed for AI.

Level 3: Optimized

Adding AI tools without changing the processes around them will hold you back.

Now its about your processes, not the tools

- Are your processes designed to take advantage of the new capabilities?
- Do you have the safety harnesses in place to allow the AI and analysts to go faster?

Is your organization optimized for the new processes?

- What roles go away? What roles will you need more of?
- Can you keep up with the speed of attacks?

How are you protecting your AI agents?

- These are now critical infrastructure, how are you protecting them?
- Do you have telemetry to know when an Agent goes bad?

Think big: If you had to start from scratch now, how would you do it?

Optimizing your Processes for AI

Question everything!

Which of your old processes are becoming bottlenecks to performance?

- Are there steps where you no longer require human action?
- How do you manage your SIEM? Can it be more more dynamic?
- What safety guardrails can you add to the AI to accelerate?
- “Because this is how we’ve always done it” isn’t a good answer.

Optimizing your organization for AI

This is the scary one

Which traditional roles are no longer needed?

- Are there roles where you no longer require human action?
- What does an entry-level SOC position look like in 5 years?
- What does your job look like in 5 years? 10 years?

Which roles will we need more of?

- The attacks aren’t going to stop
- AI doesn’t change the fundamentals of security

How do you protect your agents?

Attackers are already looking at this

How are you watching for prompt injection?

- Can you trust an agent to do this?
- Deterministic code can help, but attackers are sneaky.
- You look for ‘instructions’ and they send ‘insrutcions’
- Agents understand typos in multiple languages, good luck

What happens if your Agent goes rogue?

- Do you have the telemetry to monitor agent activity?

How can I ever trust AI to close an alert?

Spoiler: you have been doing this for years

Modern EDR has been using AI for a while
Trained specialized malware models have evaluated binaries on your endpoints for years

You already grant it authority
AI decides when to quarantine and delete files on live workstations, autonomously

So how different is letting it close a SIEM alert?

What does the future look like?

- **GPUs have gotten ~10x faster every decade, Memory gets faster and bigger**
  What happens when compute and ram isn’t the bottleneck?
- **By 2036, one GPU = ten RTX 5090. In 2046 one GPU = 100 RTX5090**
  Frontier-class inference on every desk and in every hand
- **What happens when a workstation runs today's frontier models?**
  On-prem inference looks better
- **And what do the models themselves look like in 2036? In 2046?**
  Build your Agentic SOC to be modular, the only constant is change

Thank you

Several blogs on this topic: **elastic.co/security-labs**

Scan for Elastic Security Labs

Aaron Jewitt · Detection Engineering & Security Operations, Elastic