OSS SUMMIT KOREA 2026 · AUGUST 12 · ORCHID 2

TFDrift-Falco

Terraform drift, the moment it happens — and who did it.

Real-time Terraform drift detection, powered by Falco

Keita Higaki @keitah0322 personal OSS · MIT · v0.14.0

Personal project. Views are my own — not a Sysdig product or position.

deck v1.18 · 2026-08-11

WHO I AM

Keita Higaki

Senior Customer Solutions Engineer · Sysdig

Japan AWS Top Engineer 2023 · 2024 · 2025

Google Cloud Partner Top Engineer 2024 · 2025

Day to day, I'm inside other people's cloud accounts. Securing them — and being there when something goes wrong.
Which means I read a lot of Terraform.

So none of this is a lab result. It's the thing I kept running into.

PERSONAL OSS

TFDrift-Falco

detect — what changed, and who

whyq

understand — why it matters

remedify

the exact command that fixes it

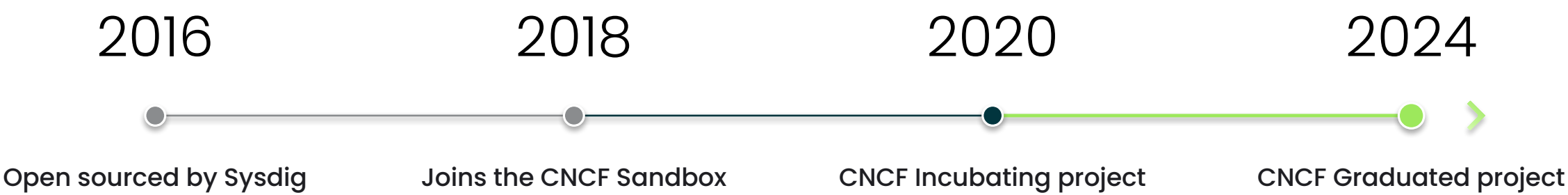
@keitah0322

> **Everything here is personal work. Not a Sysdig product, and not a Sysdig position.**

FALCO, SO FAR

A decade of detection.

From a kernel-tapping experiment to a CNCF graduated project.



Downloaded over 100 million times. Amazon, Apple, IBM and Red Hat contribute.

> **Ten years old, and graduated. The engine underneath this project is not the experimental part.**

THE ENGINE

Falco, in one slide.

If you haven't used it: Falco watches what is actually happening on a machine, and tells you when something looks wrong.

01

events

things that happen, as they happen

02

rules

plain YAML, checked against every event

03

alerts

the fields you asked for — stdout, HTTP, Slack

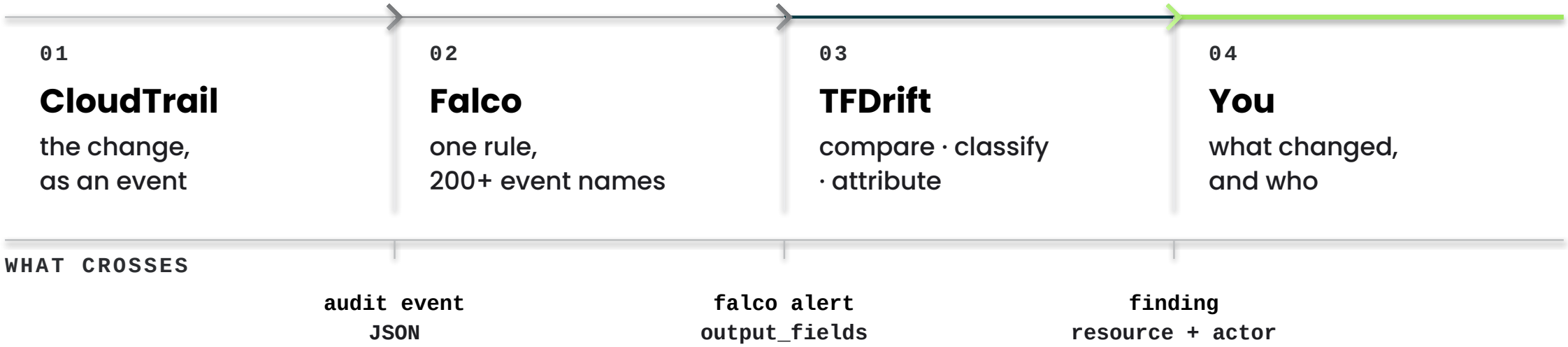
TWO WORDS FOR LATER

`syscall`	The events normally come from the kernel: a process started, a file opened, a port opened.
`plugin`	But the event source is swappable. Falco calls that a plugin — and one for CloudTrail already existed.

> **A rules engine over a stream of events. Nothing in that sentence says "syscalls".**

THE WHOLE THING

One event, four stages, one answer.



The rest of this talk is how each stage works.

> **Static drift detection alone doesn't stop an incident. So I made it real time.**

PROBLEM

Terraform was the source of truth.

```
$ terraform plan
~ aws_security_group.web
    ~ ingress { cidr_blocks = ["10.0.0.0/8"] -> ["0.0.0.0/0"] }
Plan: 0 to add, 1 to change, 0 to destroy.
```

Who opened it? When? Why?

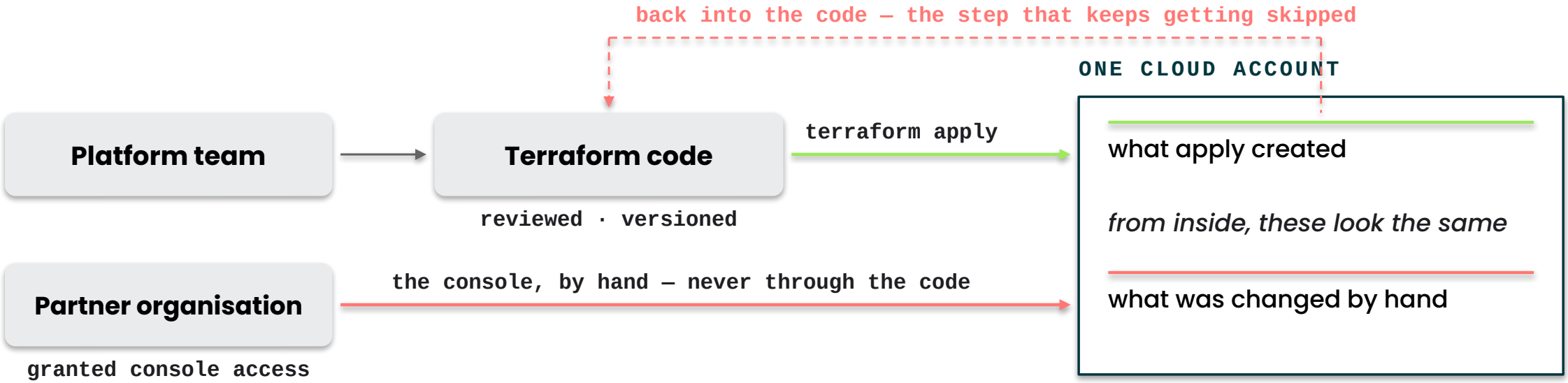
plan doesn't say. And no one sees it until the next person happens to run it.

> **Drift isn't the problem. Drift that nobody noticed is.**

AFTER ADOPTION

Building it is a project. Keeping it is an operation.

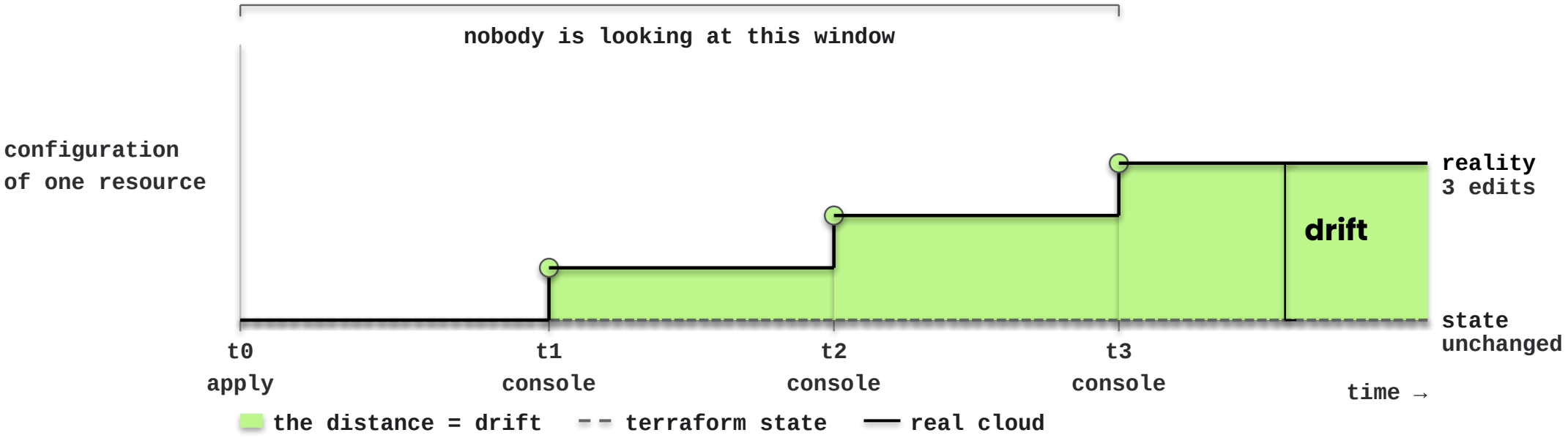
Two organisations, one account — and only one of them goes through the code:



In the case that started this project, one incident traced straight back to that missing arrow.

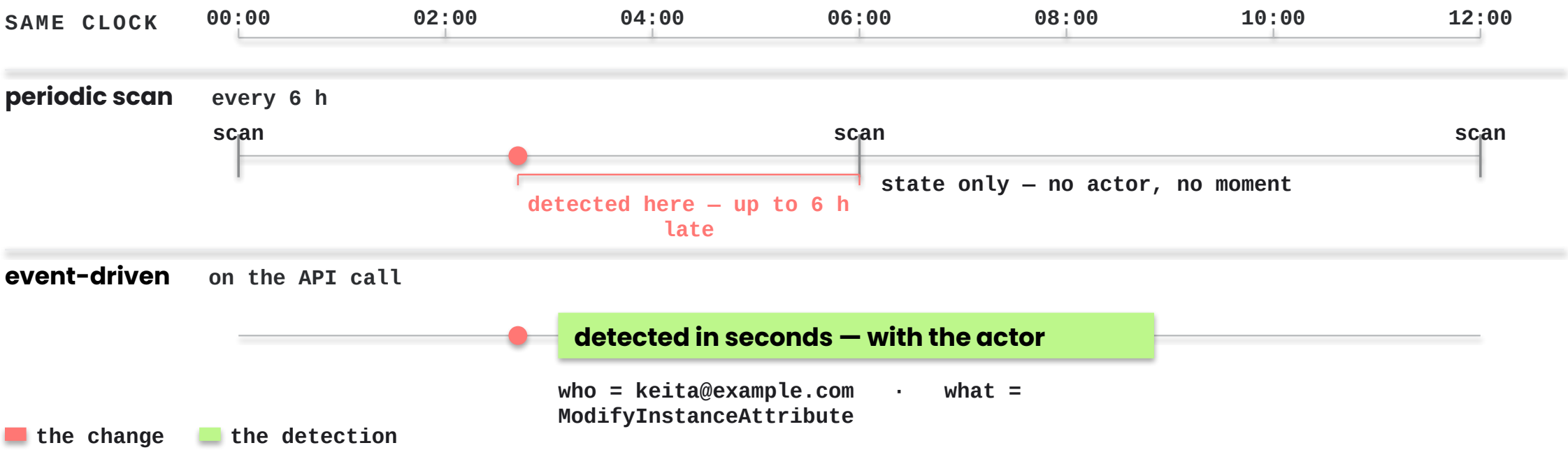
> **The organisation that made the change is not the organisation that owns the code.**

Two lines that stop agreeing.



> **Drift is not an event. It is a distance that grows while you look away.**

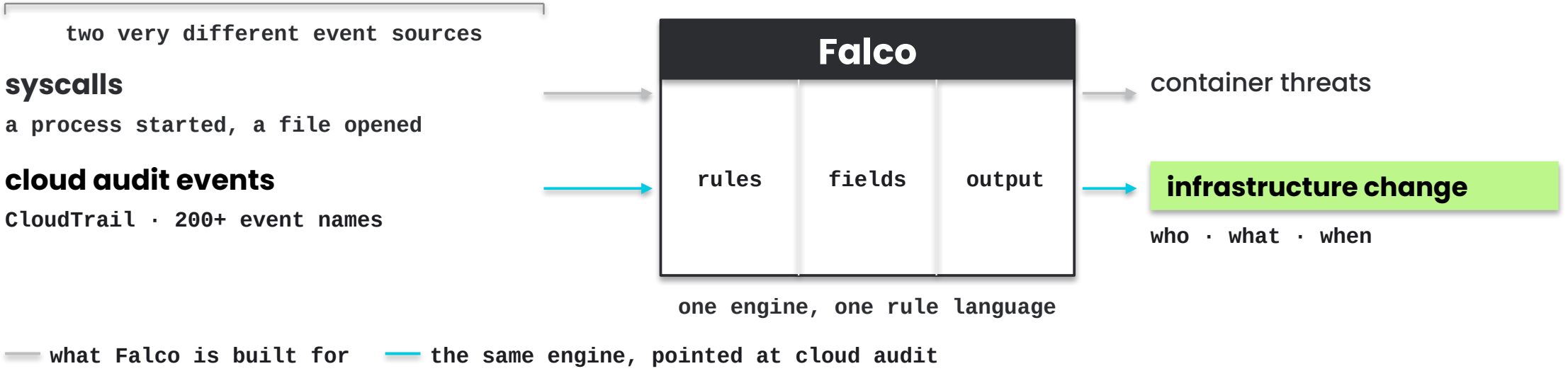
Periodic scans lose the moment — and the actor.



> **What you want isn't a snapshot of state. It's the change itself.**

SOLUTION

Falco, pointed somewhere else.



> **Drift detection inherits everything Falco already does well.**

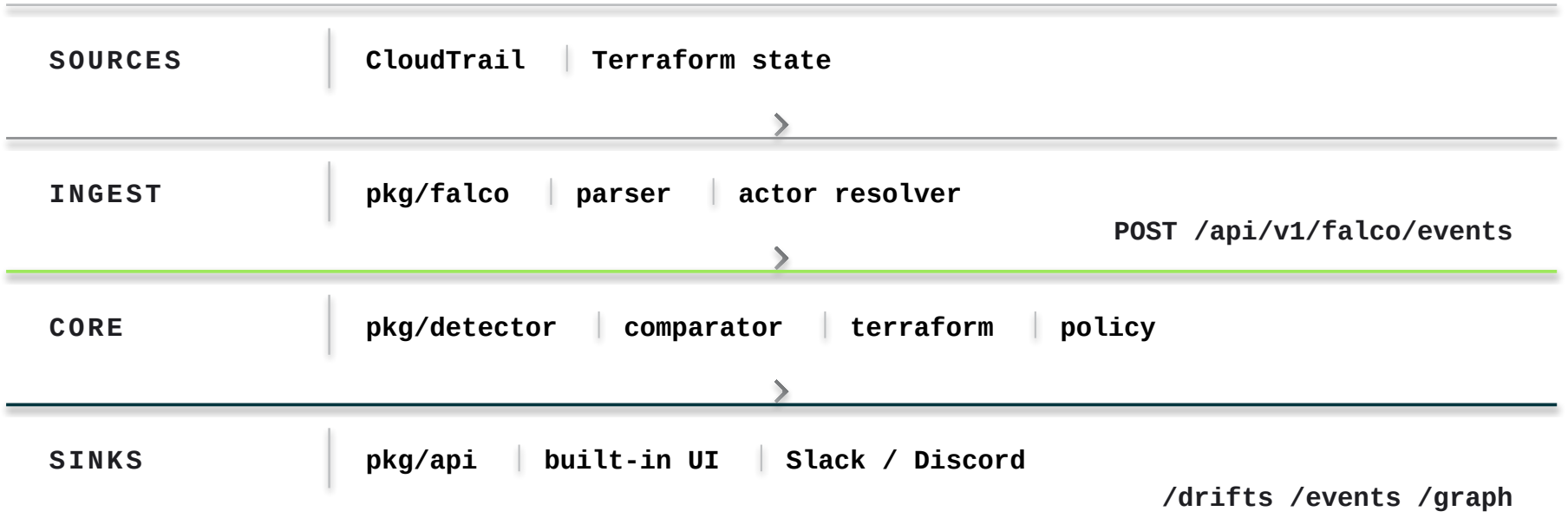
01

How it works

Four layers, one event path, and where the actor comes from.

ARCHITECTURE

Four layers, and the packages.

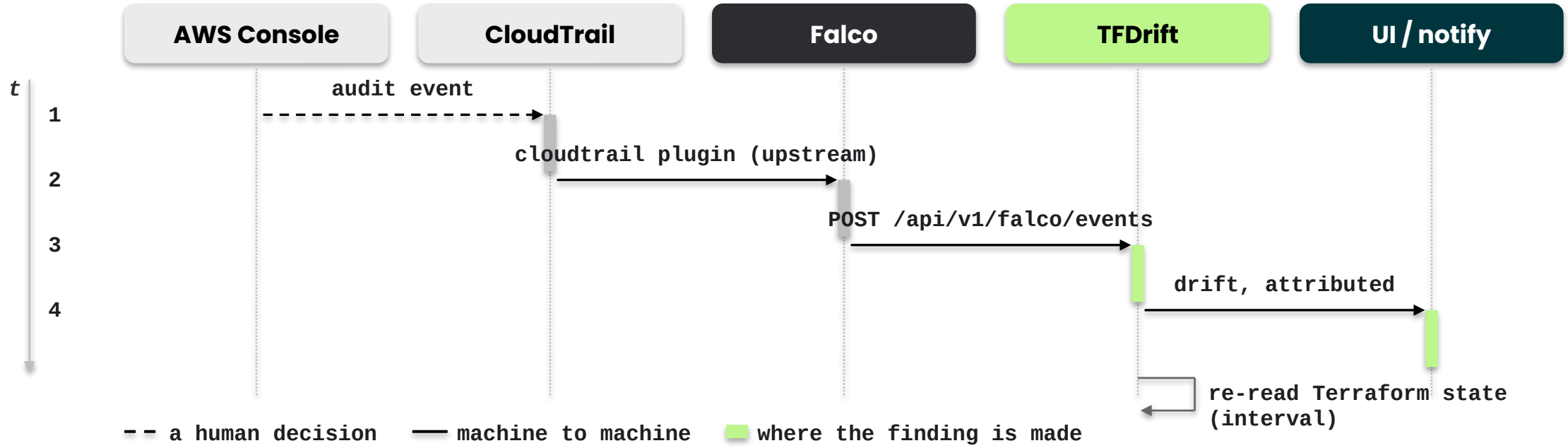


Falco stops here
nothing above or below knows it exists

cloud-agnostic
no AWS type crosses into the core

Falco is confined to the ingest layer. The core knows nothing about any cloud.

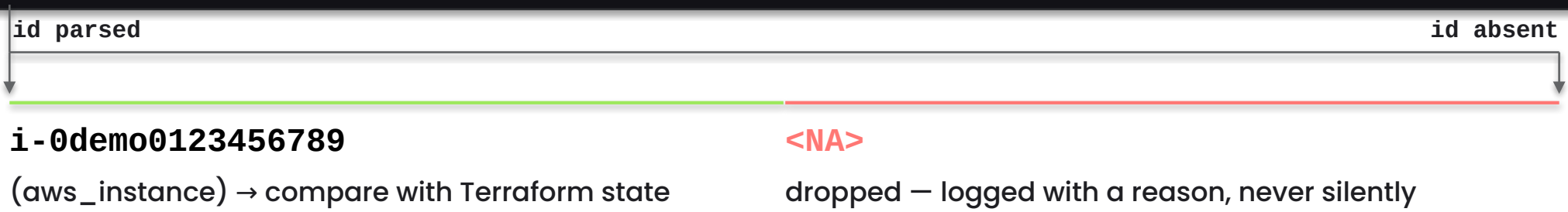
One click, four hops.



It rides Falco's own plugin and output paths — no custom protocol, no fork.

No resource ID, nothing to compare.

```
output_fields:  ct.name · ct.user · ct.request      # one aggregated field
ct.request = {"instanceId":"i-0demo0123456789"}     → parse, case-insensitive
```



CloudTrail hands over a request blob, not a resource ID. Identifying the resource is a parsing step.

> **A detector that drops findings quietly is worse than no detector.**

assumed-role hides the human.

```
userIdentity.type = AssumedRole

arn:aws:sts::123456789012:assumed-role/AWSReservedSSO_Admin_abc123/keita@example.com
```

role — not a person

session name — the human

actor = **keita@example.com**

resolved from sessionContext — not the role name

Plain IAM users are easy. SSO is the common case — and the one that erases the name.

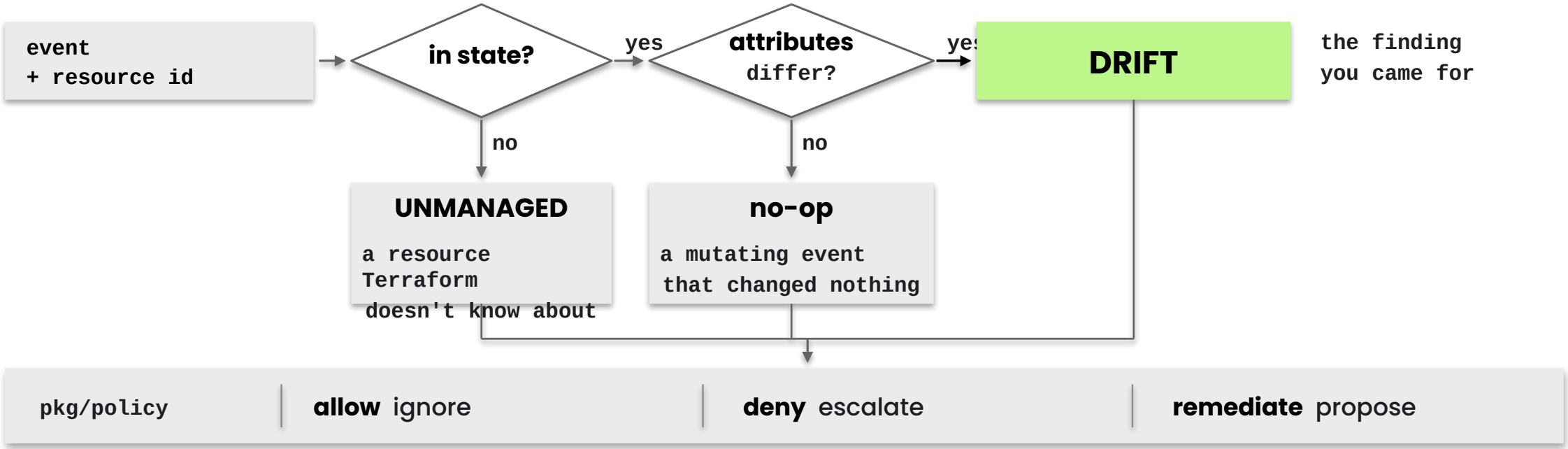
> **If SSO changes show up as a role name, it isn't an audit trail.**

IAM user

SSO / assumed-role

SSO hides it

How an event becomes a finding.



Three outcomes, one gate. Applying a fix is always separate and opt-in.

02

Does it actually work?

A live run, an honest status, and how a stranger can check.

And exactly which rule, in under four seconds.

```
$ tfdrift scan            # live AWS, ap-northeast-1
Modified (attribute differences):
  ~ aws_security_group tfdrift-demo-web (sg-...)
      ingress: terraform=[tcp|443|443|10.0.0.0/8]
               actual=[tcp|22|22|0.0.0.0/0, tcp|443|443|10.0.0.0/8]
real 3.8s   exit 0
```

one line of difference

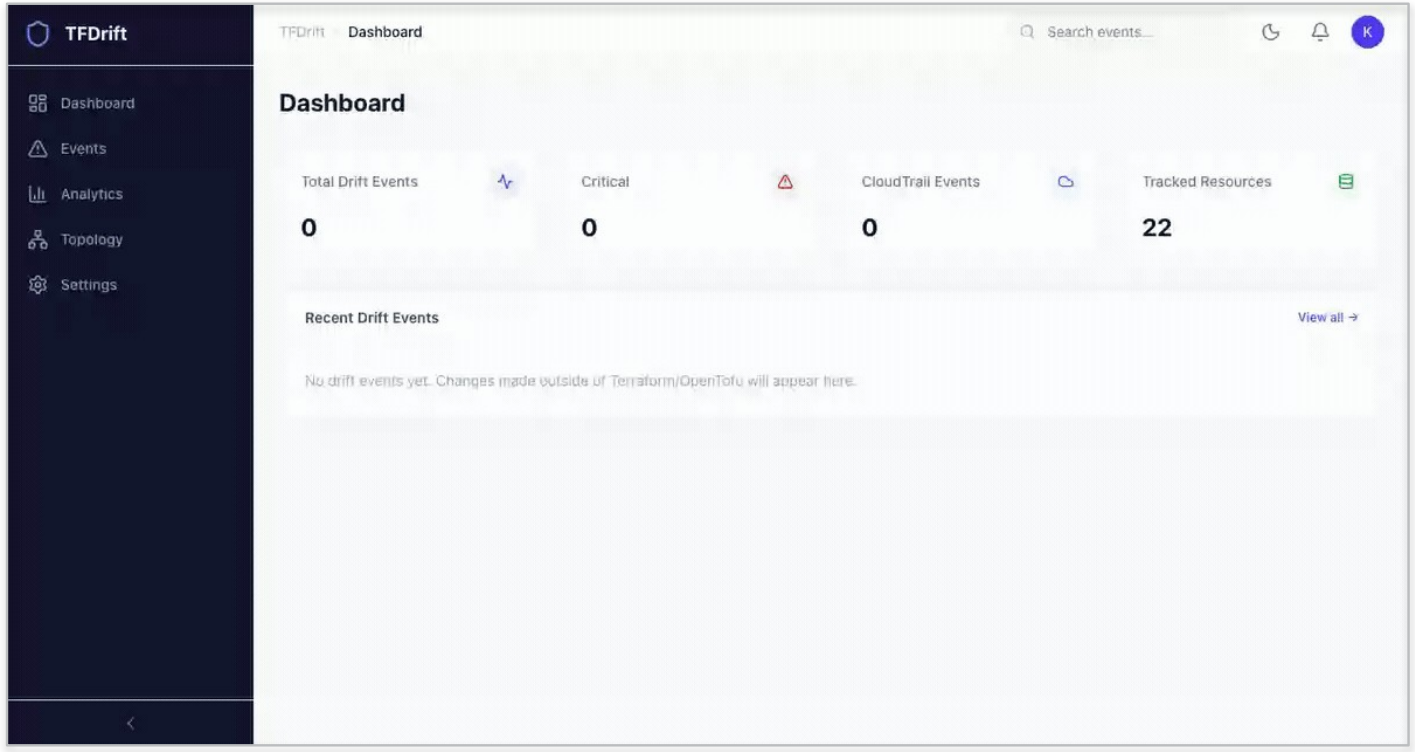
the code says 443 from the internal network. The cloud says 22, from anywhere.

3.8 seconds

state read, cloud read, compared — no waiting for a scan window.

> **That is exactly what changed. The next question is who.**

And the name attached to it.



> **Not "an alert fired" — which resource, and the human behind the role.**

WHAT TO LOOK AT

who

`vendor-dev@partner.example`

the partner organisation from that diagram — not a role name

when

`seconds after the API call`

no scan window to wait for

what

`resource modified out-of-band`

coarse on purpose — you just saw the exact diff

ABOUT THAT DEMO

Why the live act needs no cloud.

01
receiver
the real HTTP endpoint

02
parser
the real ct.request parsing

03
detector
the real compare + classify

04
UI
the real broadcast to the dashboard

WHAT I TOOK OUT, AND WHY

- **No AWS session** — `an expired token on stage proves nothing about the tool`
- **No S3 delivery wait** — `CloudTrail latency is AWS's variable, not my design`
- **The exact Falco JSON** — `byte-for-byte what Falco emits over its own HTTP output`

> **I removed the parts that fail on stage. I did not remove the parts under test.**

HONEST STATUS

What works, what doesn't.

Works

`shipping in v0.14.0`

- Event-driven detection, actor identity, resource attribution
- Classification + policy gate; reconcile (tfdrift scan)
- Drift in the built-in UI
- Webhooks: Slack, Teams, custom endpoints — retried with backoff
- An end-to-end test that runs in CI without AWS

Not yet

`alpha — the actual edges`

- The API has no authentication — it binds to localhost
- Multi-account: not tested yet
- GCP and Azure: tested, with fewer resource types than AWS
- Resource coverage is a focused set of types, by choice

> **A security tool that hides its gaps loses trust all at once.**

TRUST

Verifiable by a stranger.

RUNS IN CI ON EVERY PULL REQUEST

01
recorded alert
a real Falco JSON fixture, checked in

02
HTTP receiver
no cloud credentials needed

03
classify
the real code path, not a stub

04
assert
resource attributed, or the test fails

`author ≠ verifier`	detection changes are checked by someone other than the author
`failures surface`	discarded input is reported with a reason, and pinned by a test

> **A demo nobody else can reproduce is a demo, not evidence.**

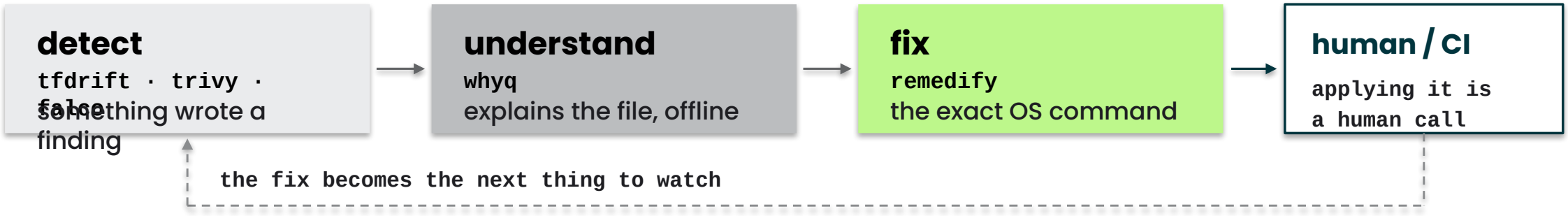
03

Where it fits

Detection is one third of the job. Then: try it.

CONTEXT

Detection is the easy part.



What travels between them is a file — Trivy, Falco, a Terraform plan, CloudTrail. Each tool runs alone, offline, and stops at the plan.
-- the loop closes here — not inside one tool

> **The deterministic tool computes. The AI only ever explains. It never invents the fix.**

TRY IT

One flag, and it starts watching.

```
$ tfdrift --auto
  zero config - it finds your Terraform state itself: local file, S3, or GCS
$ git clone https://github.com/higakikeita/tfdrift-falco   # or from source
```

MIT

license

v0.14.0

latest — OpenTofu support

AWS + GCP

CloudTrail and gcpaudit

issues

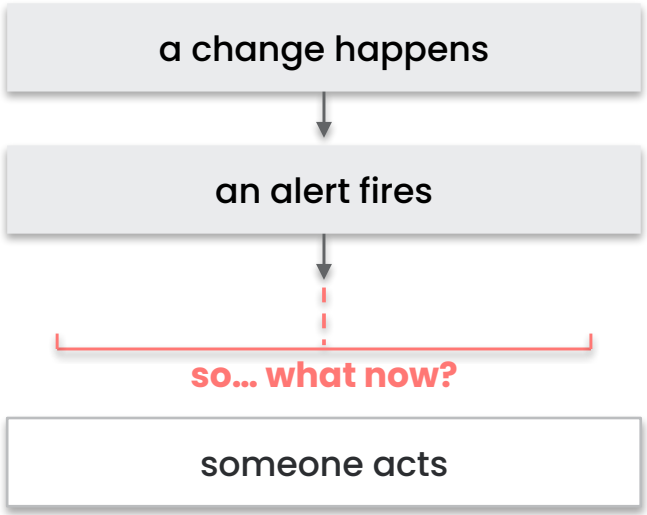
a broken input is the best bug report

> **Star it, break it, file the issue — that is what decides what I build next.**

WHY I BUILD THIS

Detection is not the goal.

THE LOOP MOST OF US RUN



Most tooling stops at the second box.
The work after it is nobody's product.

WHAT I WANT THE OSS TO DO

01 **Close the loop end to end**

detect → understand → fix, as three small open source tools anyone can run without asking me for anything

02 **Make it checkable by a stranger**

recorded evidence in CI, and the limits written down in the README — trust is built by admitting gaps

03 **Give the general parts back upstream**

the pieces that belong to Falco and the community should live there, not in my repo

> **I don't want you to adopt my loop. I want it to be cheap for you to close your own.**

Thank you

Terraform drift, the moment it happens — and who did it.

I couldn't get through all of it today. The rest is written down:

`tfdrift`	**`github.com/higakikeita/tfdrift-falco`**
`remedify`	**`github.com/higakikeita/remedify`**
`whyq`	**`github.com/higakikeita/whyq`**
`docs`	**`tfdrift-falco.vercel.app`**
`x / linkedin`	**`@keitah0322`** · **`linkedin.com/in/keita-higaki-a81377176`**

if any of them turn out useful — a star helps more than you'd think

`github.com/higakikeita/tfdrift-falco`

Questions?

Keita Higaki

`detect → understand → fix : tfdrift · whyq (MIT) · remedify (Apache-2.0)`

Personal project. Views are my own — not a Sysdig product or position.