

The riskiest feature you'll ever release is a **whole platform**.

De-Risking a Legacy CRM Migration in Life Sciences

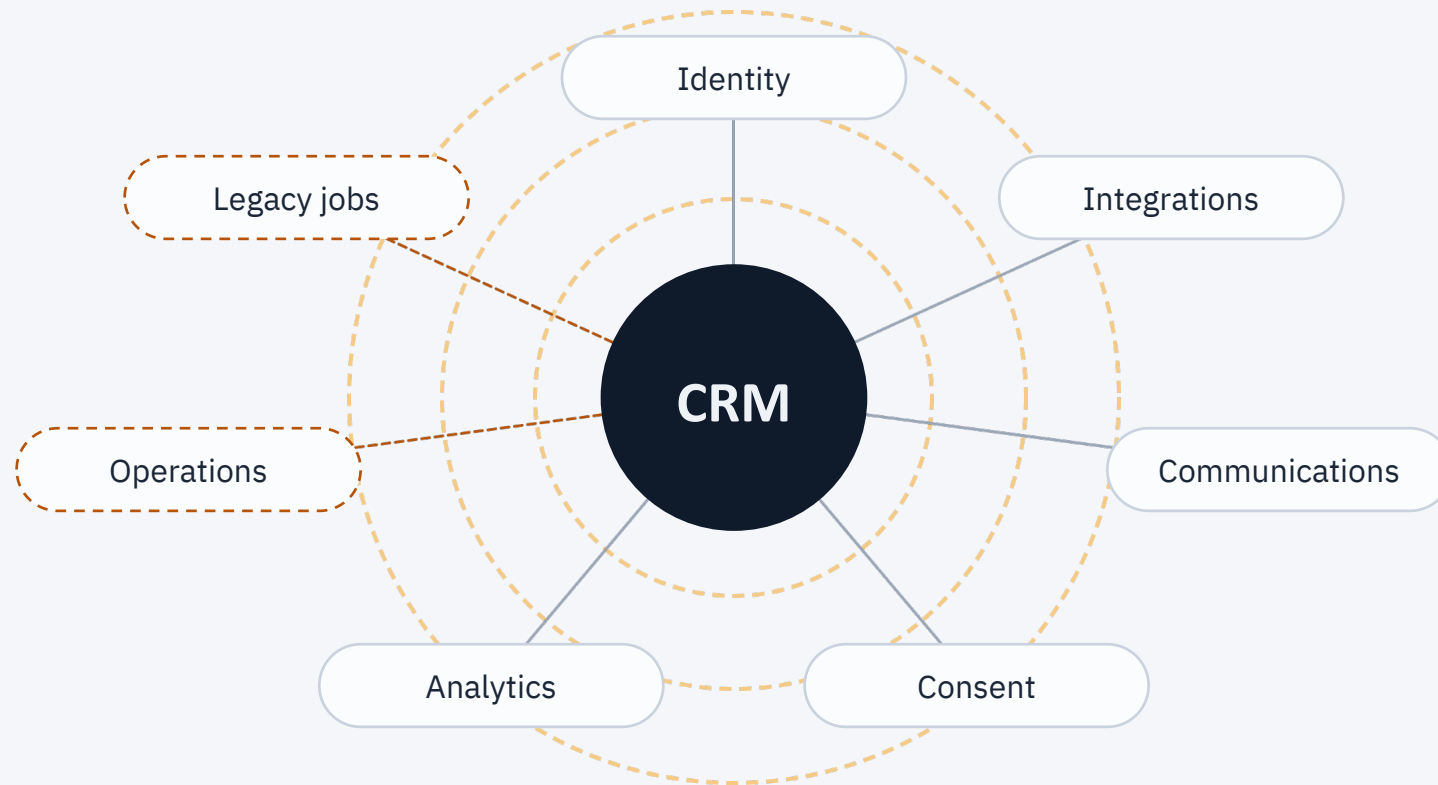
Feature-flag-driven rollout for a governed patient engagement platform



LEGACY CRM

NEW PLATFORM

Why CRM migrations fail differently



A CRM isn't an application. It's a hub.

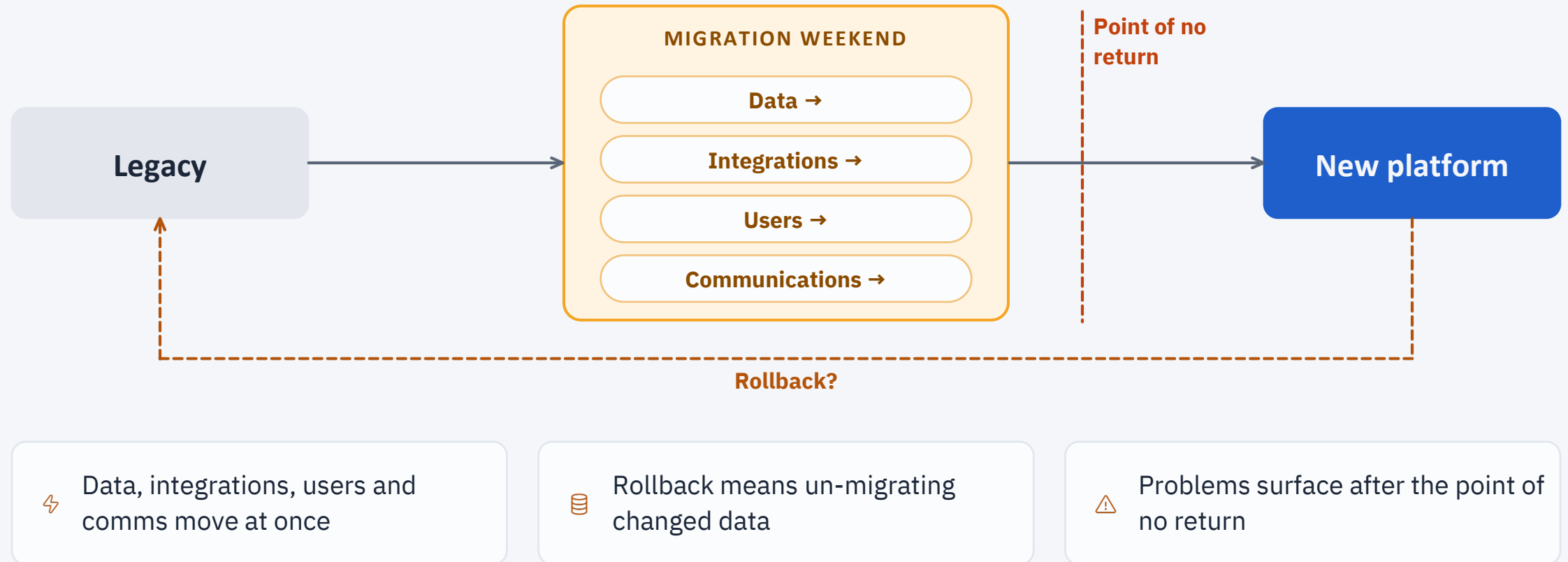
Most dependencies live in field names, nightly jobs and workflows nobody remembers.

THE REAL RISK

Blast radius

Dashed: dependencies that are rarely documented

The big-bang migration trap



What if migration behaved like a release?

Deployment

the new platform exists

DEPLOYED · configured · integrated · data loaded

Activation

people and workloads depend on it



Feature flags



Canary releases



Blue/green

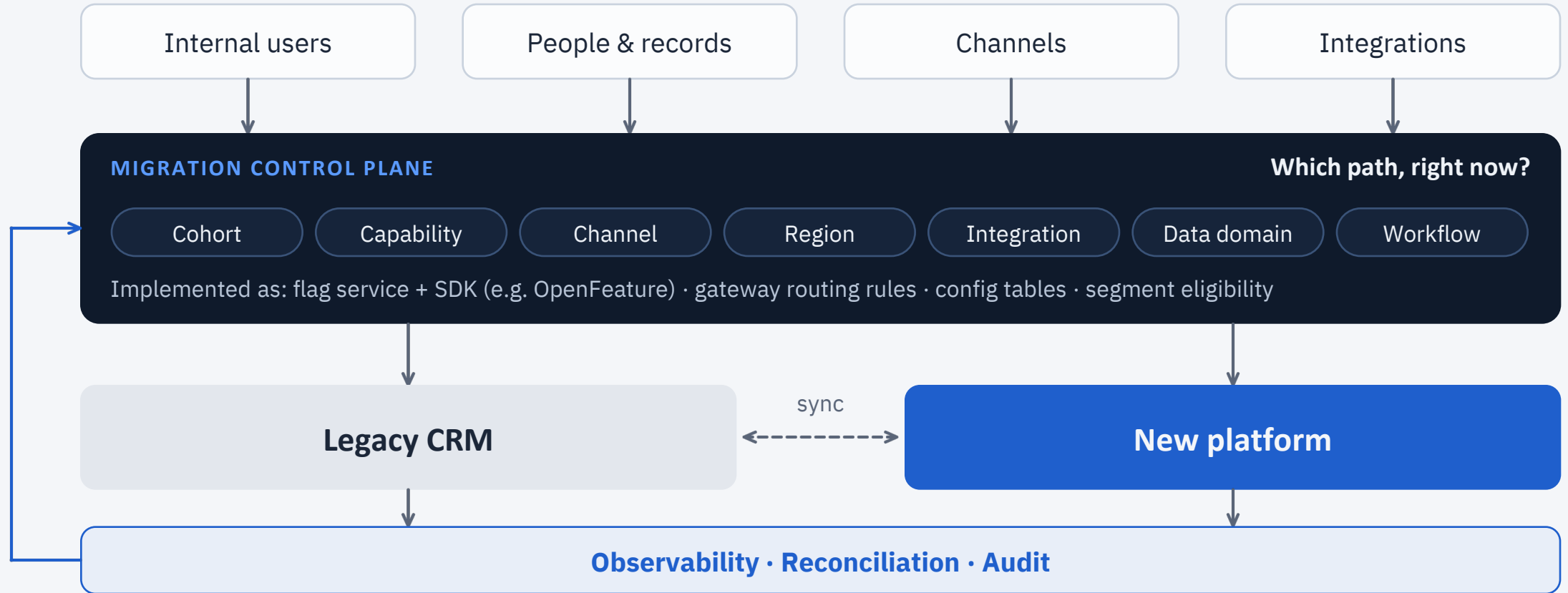


Progressive rollout

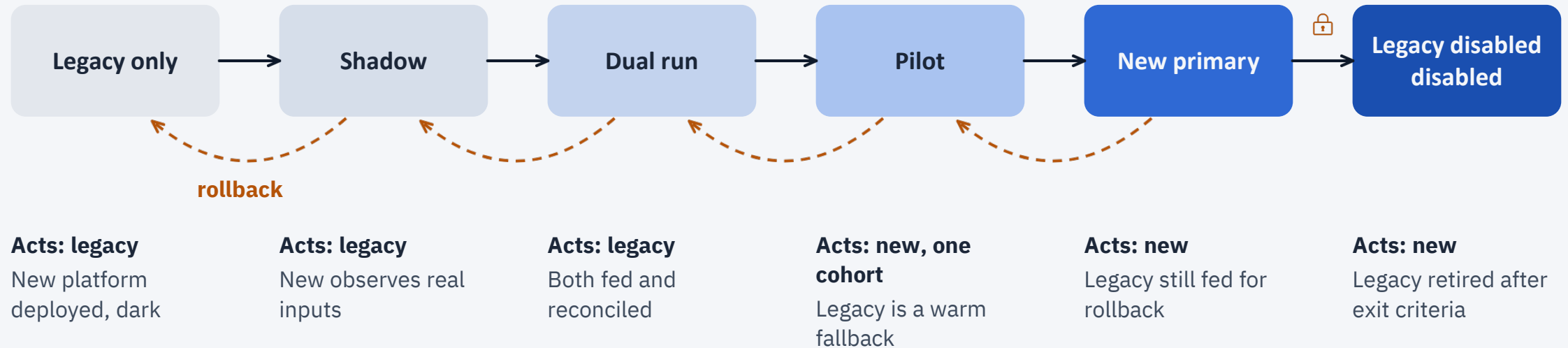
What if we applied the same separation to a platform migration?

The migration control plane

Rollout decisions live outside the code.



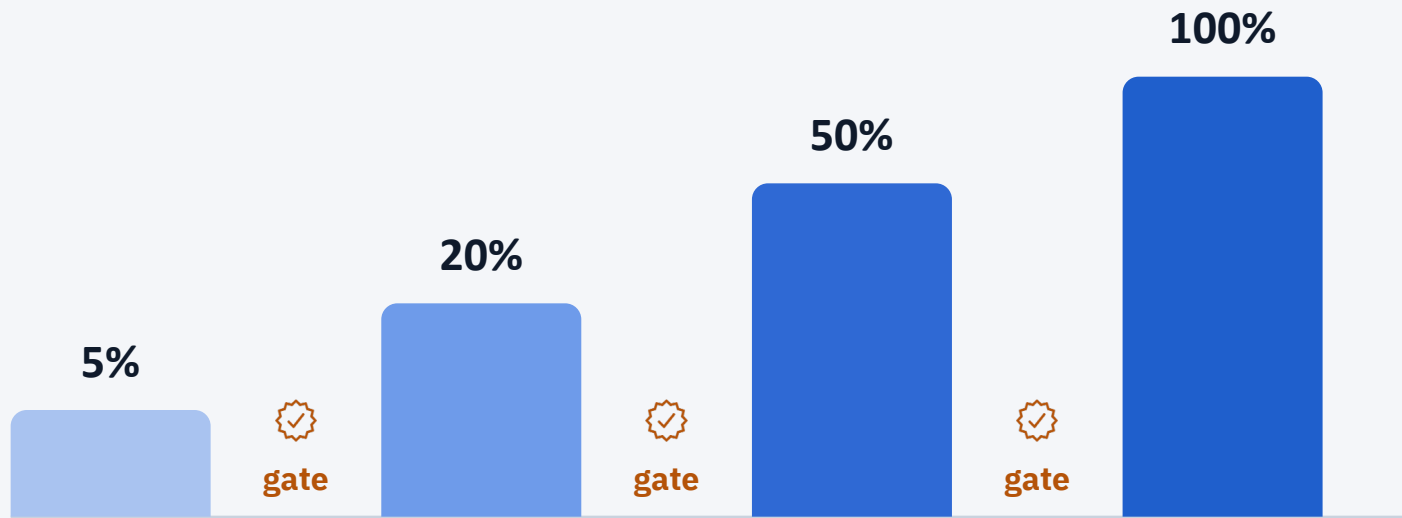
The migration state machine



Every transition: **entry criteria** + **a way back**

🔒 Only one-way door: legacy disabled

Cohort-based rollout



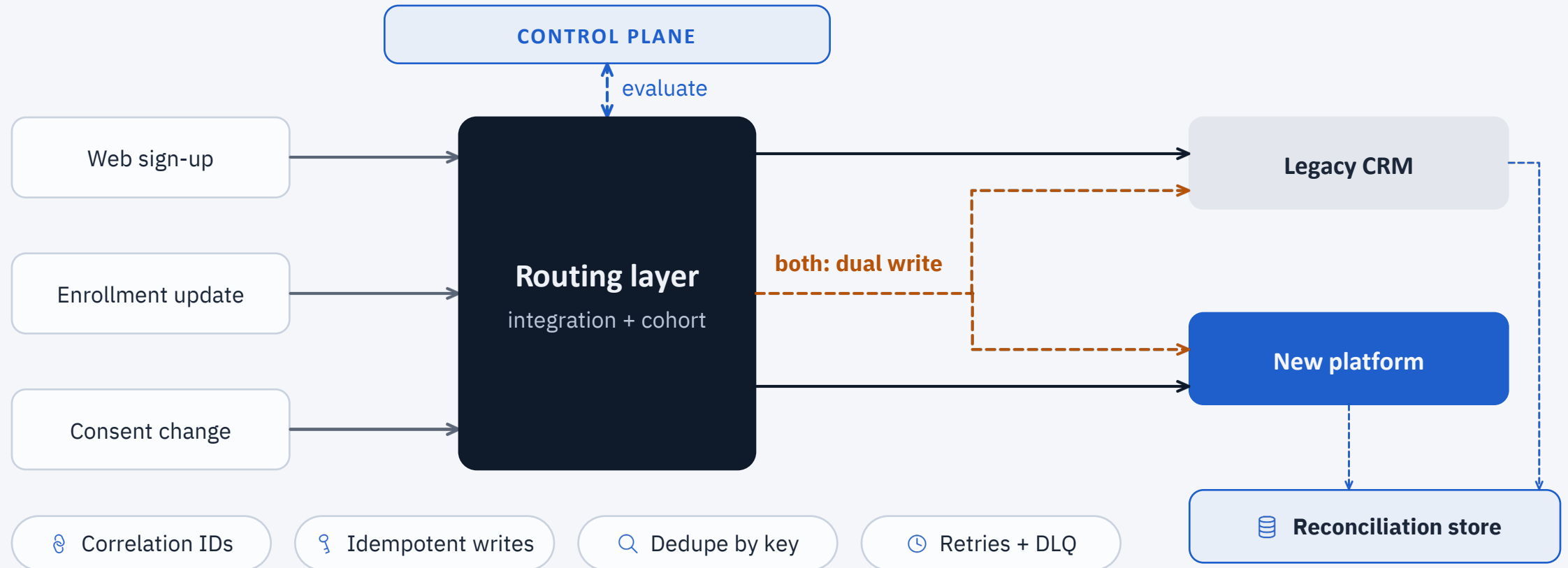
Better cohort keys

-  Team
-  Segment
-  Data complexity
-  Workflow
-  Region
-  Channel

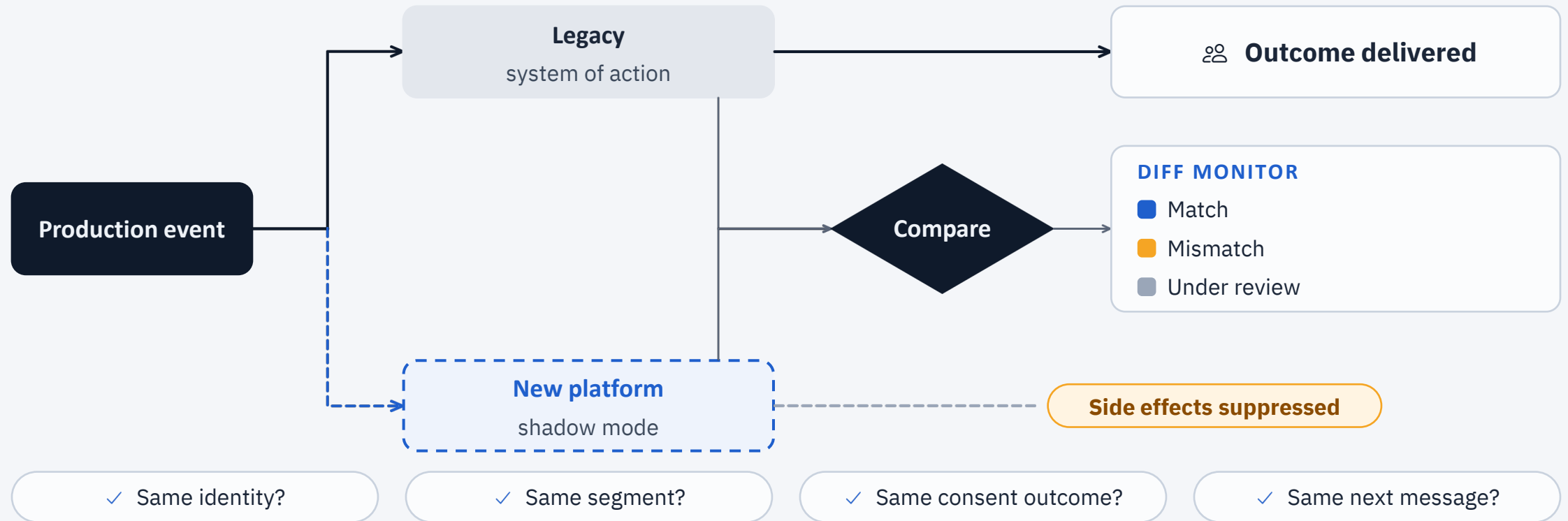


Integration routing

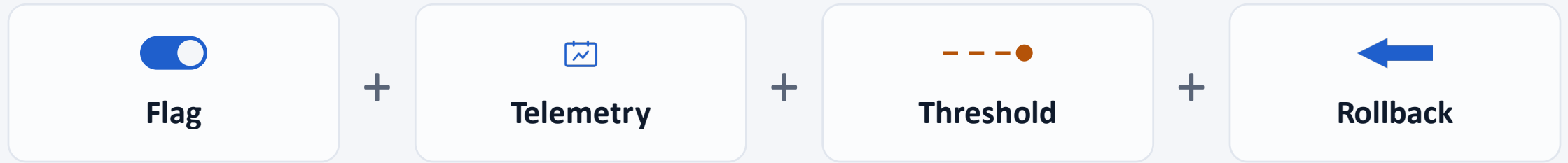
Cutover becomes a config change.



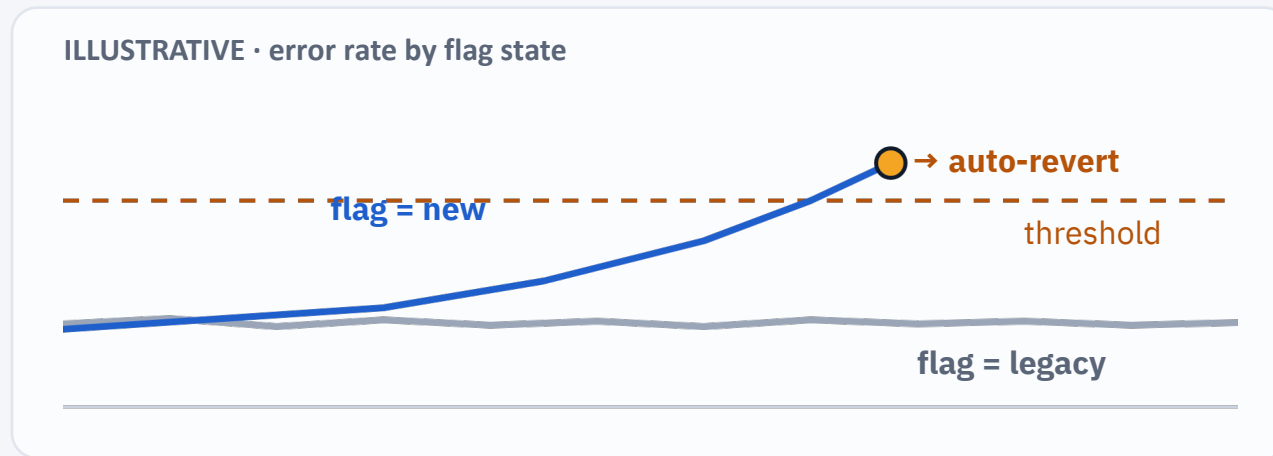
Shadow mode and dual run



Observability is part of the flag



Missing one of the four? Then you have a switch, not a control.



BEYOND SRE SIGNALS

API errors

Latency

Recon mismatches

Duplicates

Stalled workflows

Message failures

Data-quality issues

Support tickets

Regulated environments change the rules

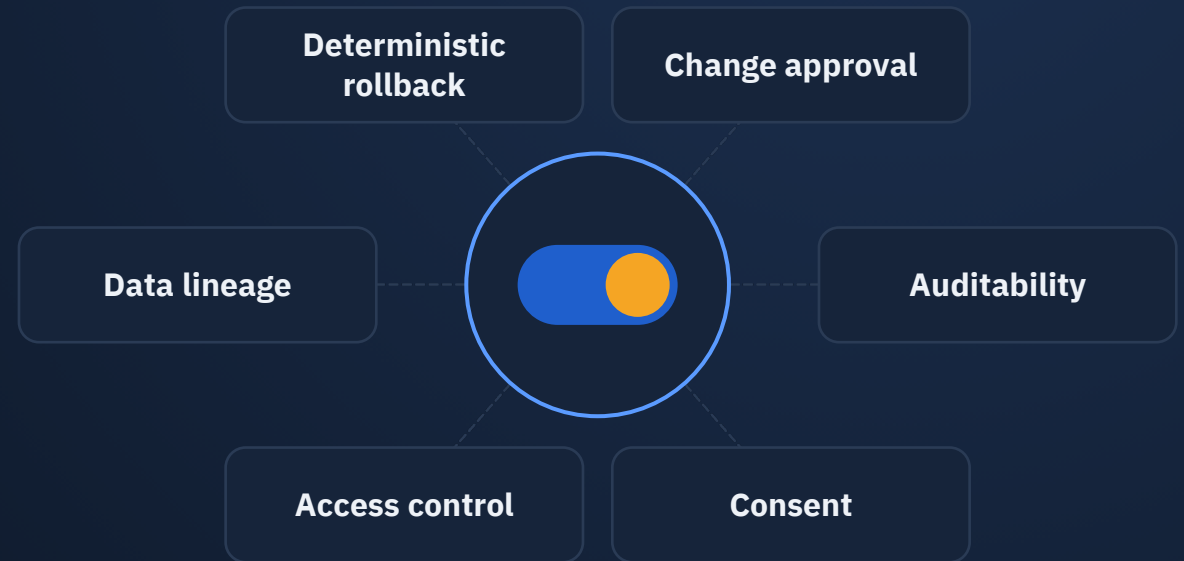
A flag change is a change to **where sensitive data flows**.

AUDIT LOG

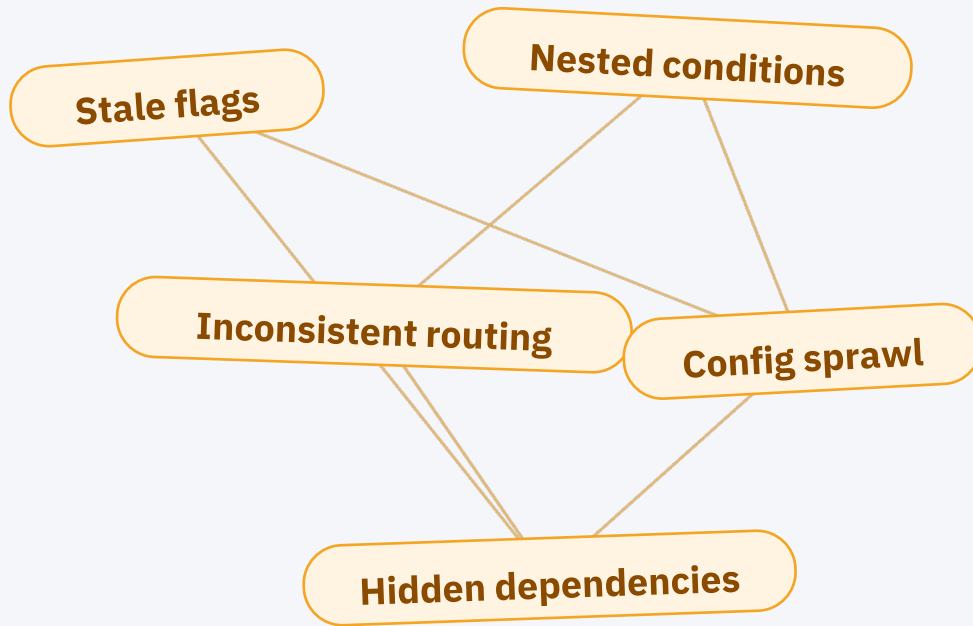
who · when · flag · from → to · change

a91f · 14:02Z · route.consent · legacy → new · CHG-20260

- ⚠ Keep sensitive data out of flag context and telemetry. Use opaque IDs.



Feature flags can become technical debt



Knight Capital, 2012: a repurposed flag reactivated retired code on one server (SEC order, 2013).

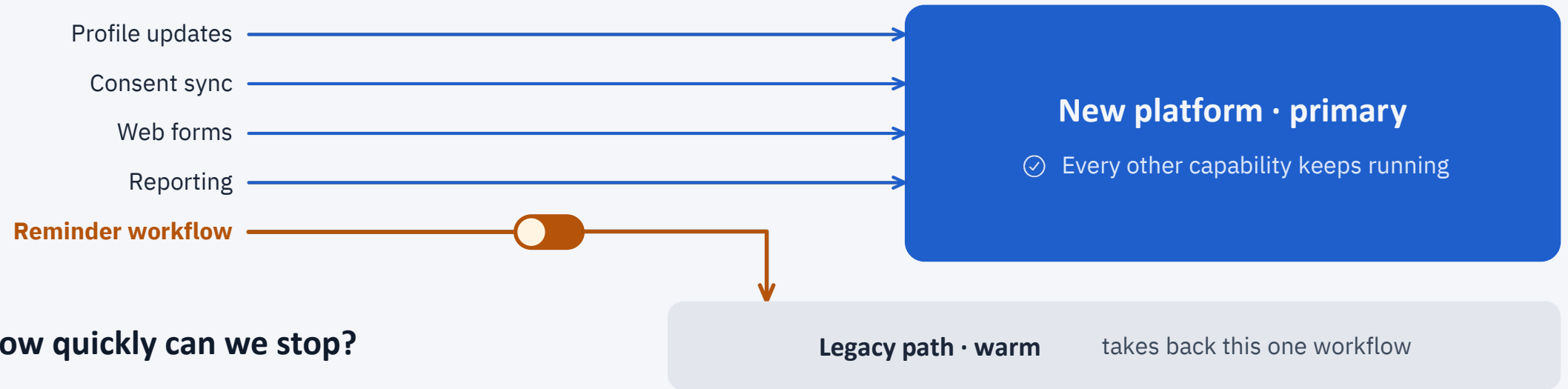
FLAG PASSPORT

migration.consent-sync

Owner	CRM Platform Engineering
Purpose	Route consent sync to new platform
Created	Week 0 of rollout
Retire by	30 days after 100% rollout
Depends on	cohort.assignment
Monitor	Mismatch rate below 0.5%
Rollback	Route back to legacy

Never reuse a flag.

The kill-switch principle



⚡ Legacy path stays warm



Kill switch is drilled, with a known propagation time



On-call can flip it without a meeting

A practical migration playbook



Platform migrations don't have to be single, irreversible events.



Profiles



Consent



Email



SMS



Web forms



Reporting



Identity



Analytics

01

Treat migration as progressive delivery

02

Use flags to shrink blast radius

03

Design rollback and observability before rollout

No single weekend should be able to hurt you.

REFERENCES

Sources and further reading

- Hodgson, P. “Feature Toggles (aka Feature Flags).” martinfowler.com
- Governor, J. et al. Progressive Delivery. IT Revolution.
- Fowler, M. “Strangler Fig.” martinfowler.com, 2024.
- Microsoft Azure Architecture Center. “Strangler Fig pattern.”
- Fowler, M. “BlueGreenDeployment”; Sato, D. “CanaryRelease.” martinfowler.com
- GitHub. Scientist: carefully refactoring critical paths.
- Newman, S. Monolith to Microservices. O’Reilly (Parallel Run).
- CNCF. “OpenFeature becomes a CNCF incubating project,” Dec 2023.
- OpenTelemetry. Semantic conventions for feature flags.
- SEC cease-and-desist order, Knight Capital Americas LLC, Oct 2013.
- 45 CFR 164.312(b). HIPAA Security Rule: audit controls.
- All examples in this talk are hypothetical.