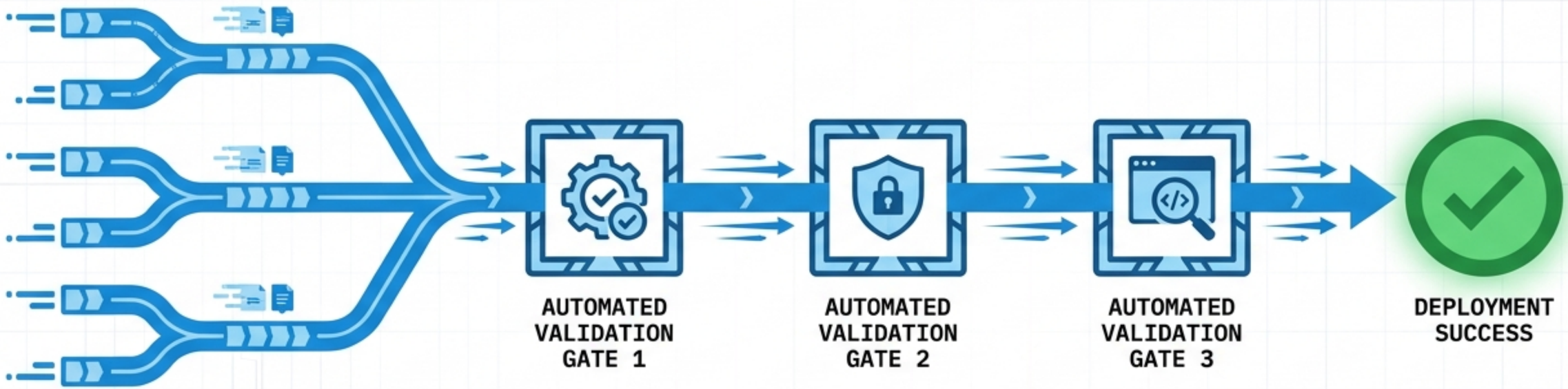
Ways of Working — S07E04: DevOps, Continuous Delivery, Trunk-Based Workflows & Deployment Safety

Moving beyond ‘CI Theatre’ to hands-off, safe, and empirical software delivery.



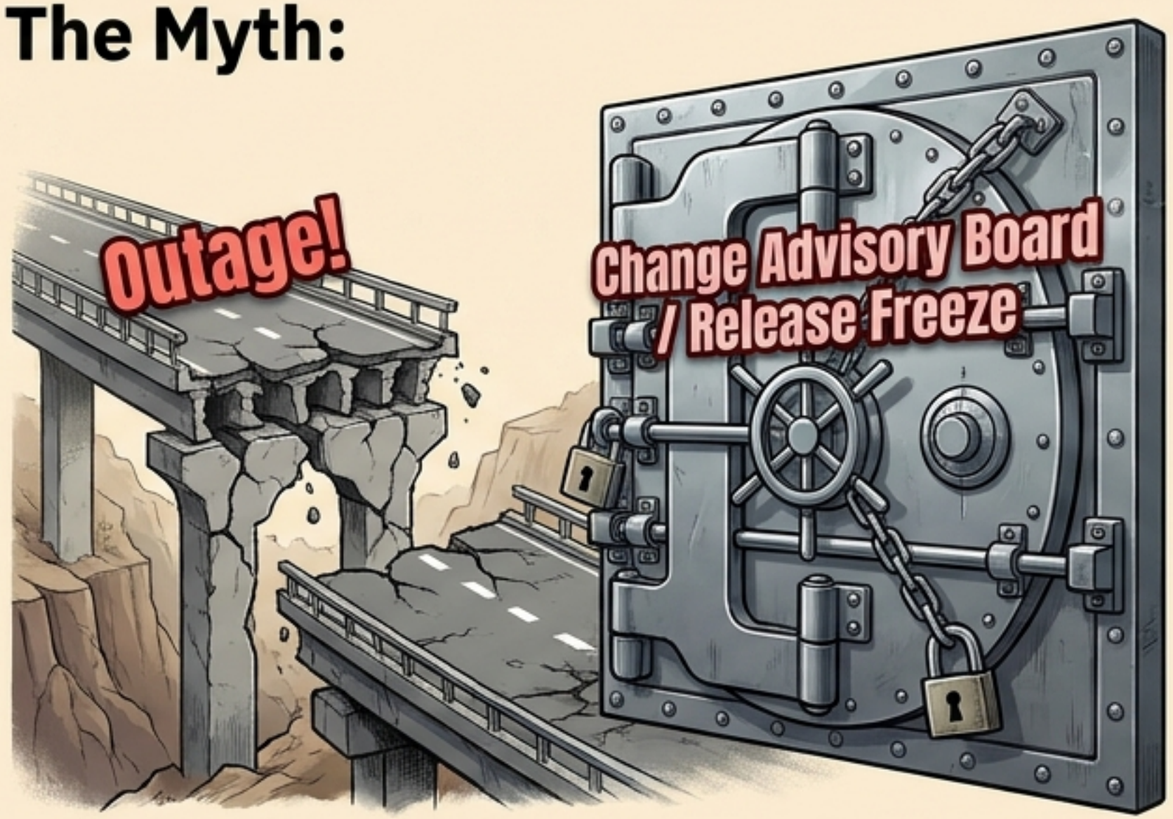
> **TARGET AUDIENCE:** Developers, Tech Leads, PMs.

> **SYNOPSIS:** Stripping away methodology dogma to look at the empirical DORA data—and the actual production pipelines used by AWS and Google SRE.

Gemini Notebook

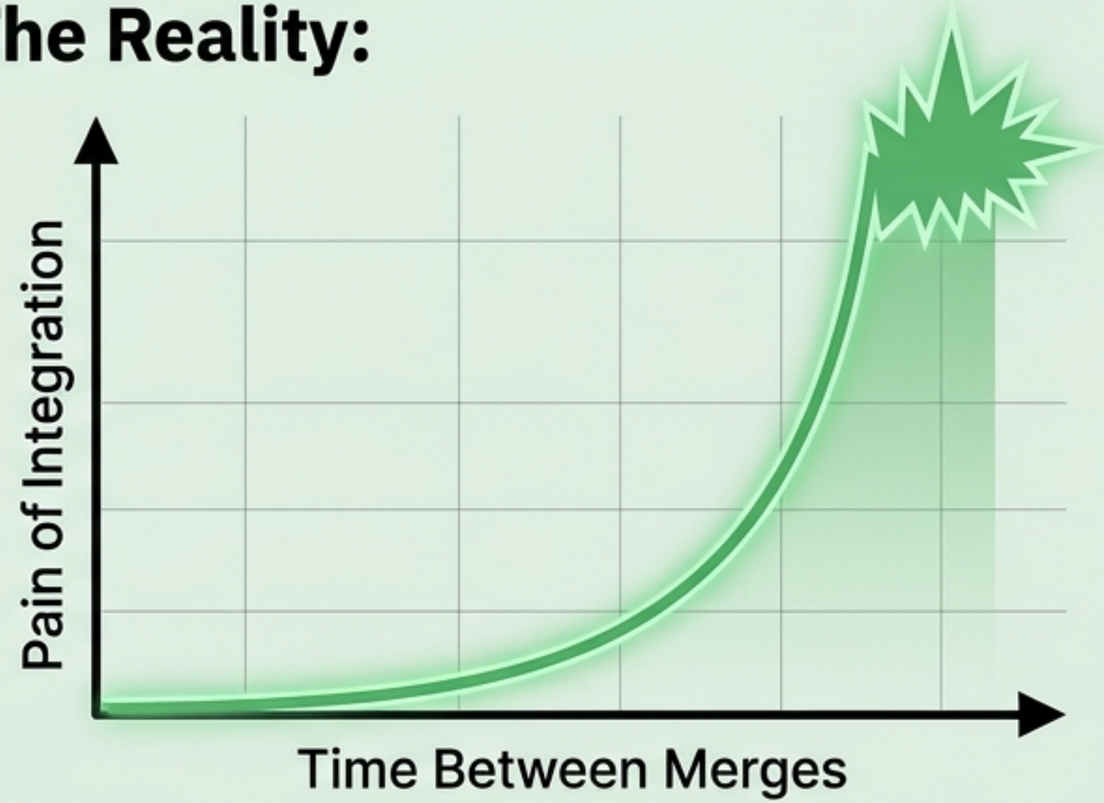
S07E03 Recap: Honest forecasting relies on small batches and flow metrics, not story-point theatre.

The Myth:



To make systems stable, we must slow down, freeze code, and add manual CAB approvals.

The Reality:



Delaying integration creates massive, terrifying release batches. Speed and stability are not trade-offs; they require each other. **If it hurts, do it more often.**

Gemini Notebook

S07E04: The DevOps Pipeline

Terminal Window

Continuous Integration (CI)

```
> STATUS: DEVELOPER WORKFLOW
```

Developers merge into the shared mainline at least daily.

Verified by an automated, self-testing build taking <10 minutes.

Terminal Window

Continuous Delivery (CD)

```
> STATUS: BUSINESS CAPABILITY
```

The codebase is in a deployable state at any given time.

Releasing to production is a push-button business decision.

Terminal Window

Continuous Deployment

```
> STATUS: PIPELINE AUTOMATION
```

Every green build that passes the automated pipeline is automatically pushed to production without human intervention.

Gemini Notebook

S07E05: Elite DevOps Performance Metrics

Speed / Throughput



How often code is successfully deployed to production



Time from commit to running in production

Stability / Reliability



Percentage of deployments requiring immediate intervention



Time to restore service after a failed deployment

A decade of empirical data proves that Elite teams excel at all four simultaneously. Small batches shrink the rollback blast radius, making root-cause isolation instant.

Gemini Notebook

S07E06: The Elite Performance Enablers

Small Batches

The catalyst for everything. Reduces cognitive load, review time, and integration friction.

Loosely Coupled Teams

Conway's Law in action; teams can deploy independently without orchestrating multi-team release trains.

Test Automation

Reliable, developer-maintained suites that only pass releasable code. Replaces manual QA freezes.

Observability

Tooling to actively debug systems and monitor predefined telemetry, enabling safe automated rollouts.

`> OUTCOME: Elite Software Delivery Performance`

Gemini Notebook

S07E07: Organizational Culture and Performance

Pathological (Power-oriented)	**Bureaucratic** (Rule-oriented)	**Generative** (Performance-oriented)
• Messengers are shot. • Low cooperation. • Low cooperation.	• Messengers are neglected. • Modest cooperation.	• Messengers are trained. • High cooperation. • Risks are shared.
Failure leads to scapegoating.	Failure leads to justice/process.	Failure leads to blameless inquiry.

DORA research empirically proves that a Generative, high-trust culture directly drives software delivery performance. Psychological safety is a technical prerequisite.

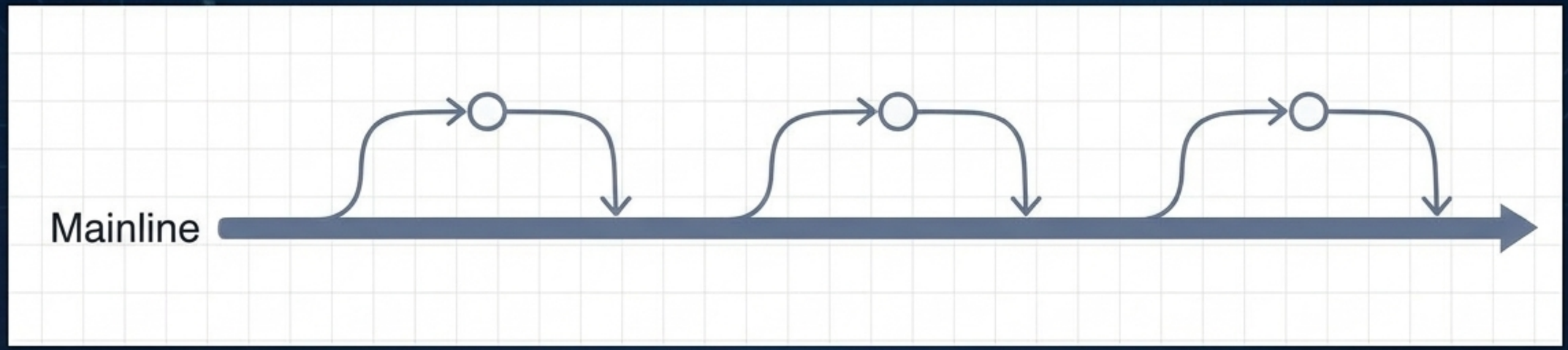
Gemini Notebook

S07E08: DORA Lead Time & Operational Friction

GitFlow / Long-Lived Feature Branches	**GitHub Flow** (Multi-Day PR Queues)	**Trunk-Based Development** (<1-Day Branches)
Integration Friction: Extreme (Semantic merge hell).	Integration Friction: Moderate (Synchronous reviews block flow).	Integration Friction: Minimal.
DORA Lead Time: Weeks / Months.	DORA Lead Time: Days.	DORA Lead Time: Hours / Minutes.
Rollback Blast Radius: Massive.	Rollback Blast Radius: Medium.	Rollback Blast Radius: Tiny (Reverted in seconds).

Gemini Notebook

S07E09: The Mechanics of Trunk-Based Flow



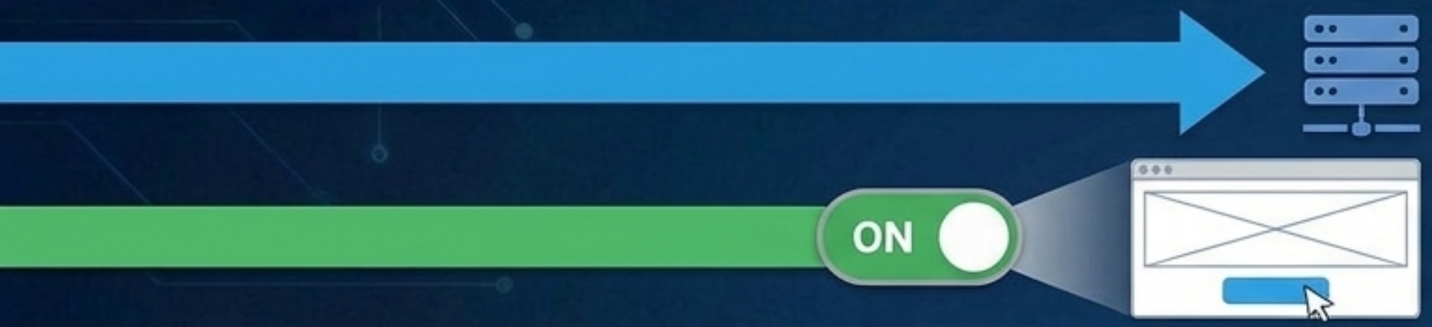
The Mechanics of Trunk-Based Flow:

```
[RULE 01] No branch lives longer than 1 day.
[RULE 02] Requires fast (<10 min), fully automated, green builds.
[RULE 03] Avoids 'CI Theatre': Integrating daily prevents the illusion
          of isolated tests that clash upon integration.
```

But what if my feature isn't finished in a single day?

Gemini Notebook

S07E10: Decoupling Deploy from Release



> **DEPLOYMENT** (Technical Event)

> **RELEASE** (Business Event)

Pete Hodgson's Feature Toggle Categories:

Release Toggles:
Hiding incomplete or untested work from end-users on production.

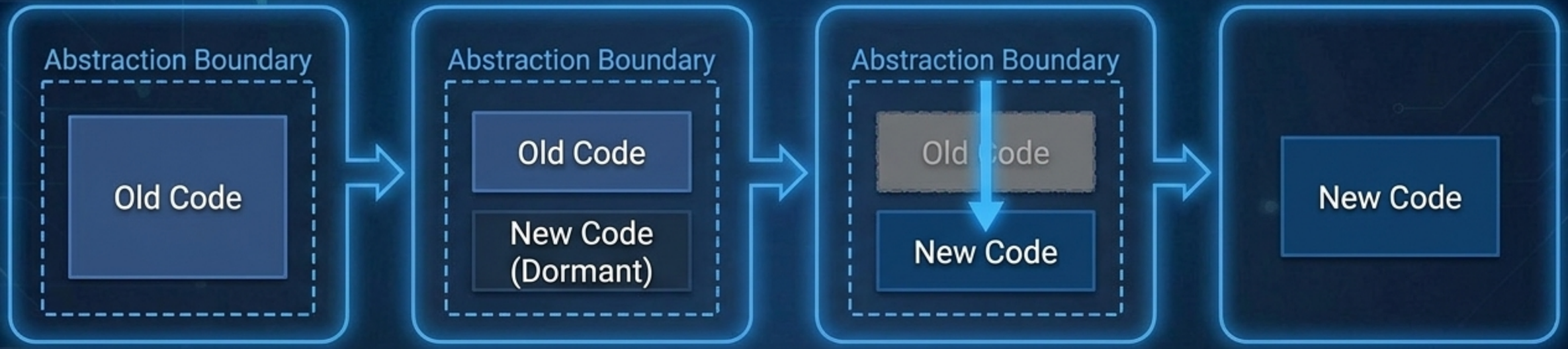
Experiment Toggles:
A/B testing product hypotheses with live traffic.

Ops Toggles:
Gracefully degrading non-critical systems under heavy load.

Permissioning Toggles:
Segmenting access for Alpha/Beta customers or internal employees.

Gemini Notebook

S07E11: Branch by Abstraction (e.g., Replacing an old ORM with a new one)



Intercept
Introduce an abstraction layer around the old code.

Build Dark
Write the new implementation alongside the old one. It is dormant, safely hidden behind the abstraction.

Flip the Switch
Update the abstraction to point routing to the new code.

Clean Up
Delete the old code, the switch, and (optionally) the abstraction.

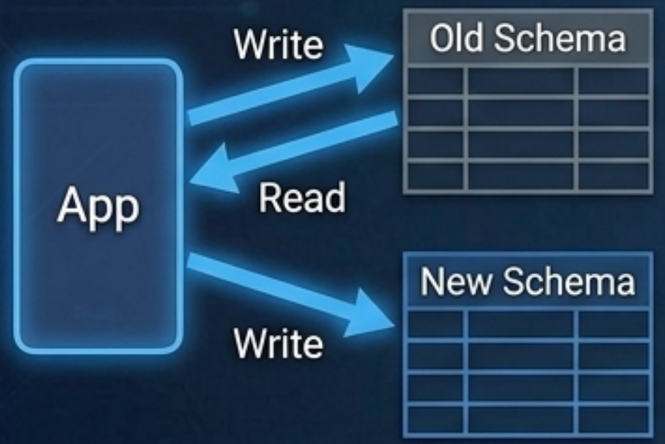
Result: The codebase remains deployable to production every single day.

Gemini Notebook

S07E10: Parallel Change (Zero-Downtime Database Evolution)

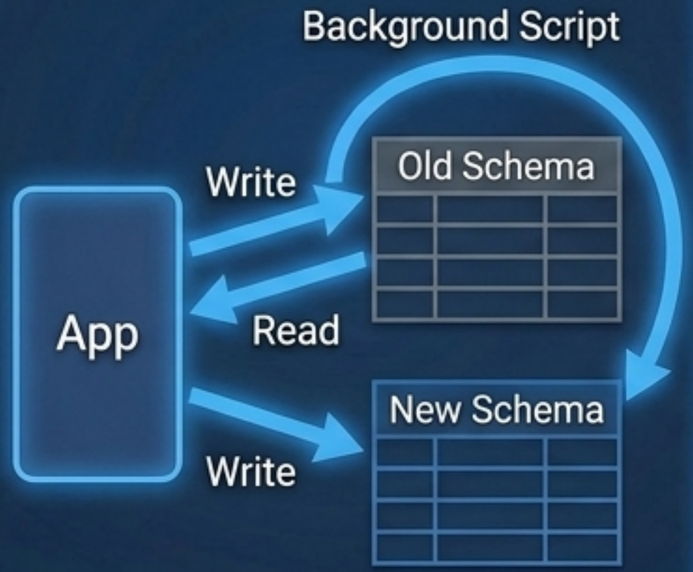
Phase 1: Expand

Add new columns/tables. The application writes to both old and new, but reads from the old.



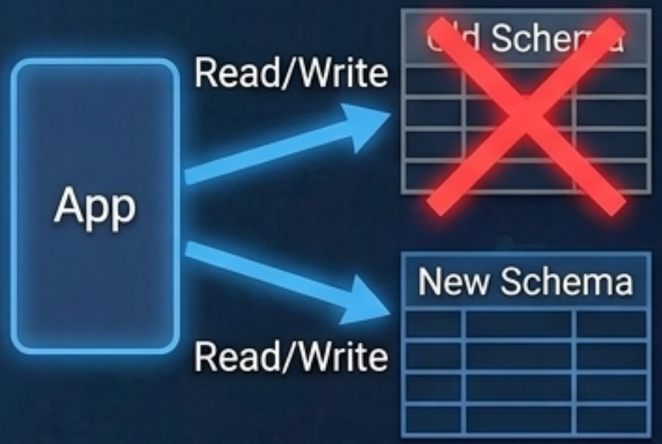
Phase 2: Migrate

Run a background script to backfill historical data from the old columns into the new ones.



Phase 3: Contract

The application shifts to read/write only from the new schema. Safely drop the old columns.



Result: Database migrations no longer block continuous delivery or require weekend maintenance windows.

Gemini Notebook

S07E12: The Progressive Deployment Pipeline



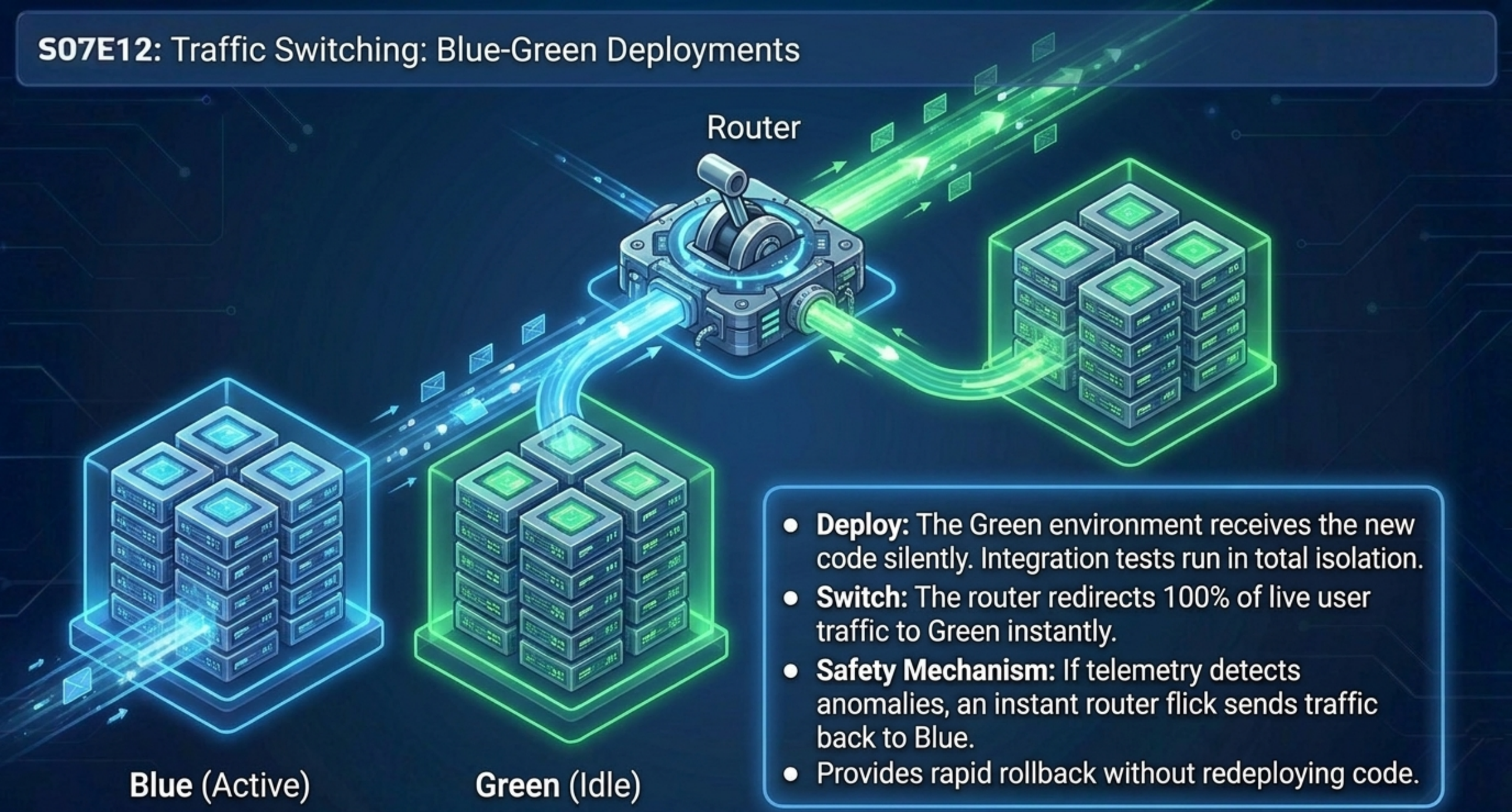
The Core Rule of Artifacts:

- Every commit generates a single immutable release candidate artifact.
- If it fails any stage, the artifact dies.
- If it passes, the exact same artifact binary moves forward.

> **FATAL ANTI-PATTERN:** Recompiling or rebuilding the code for different environments (Dev vs. QA vs. Prod).

Gemini Notebook

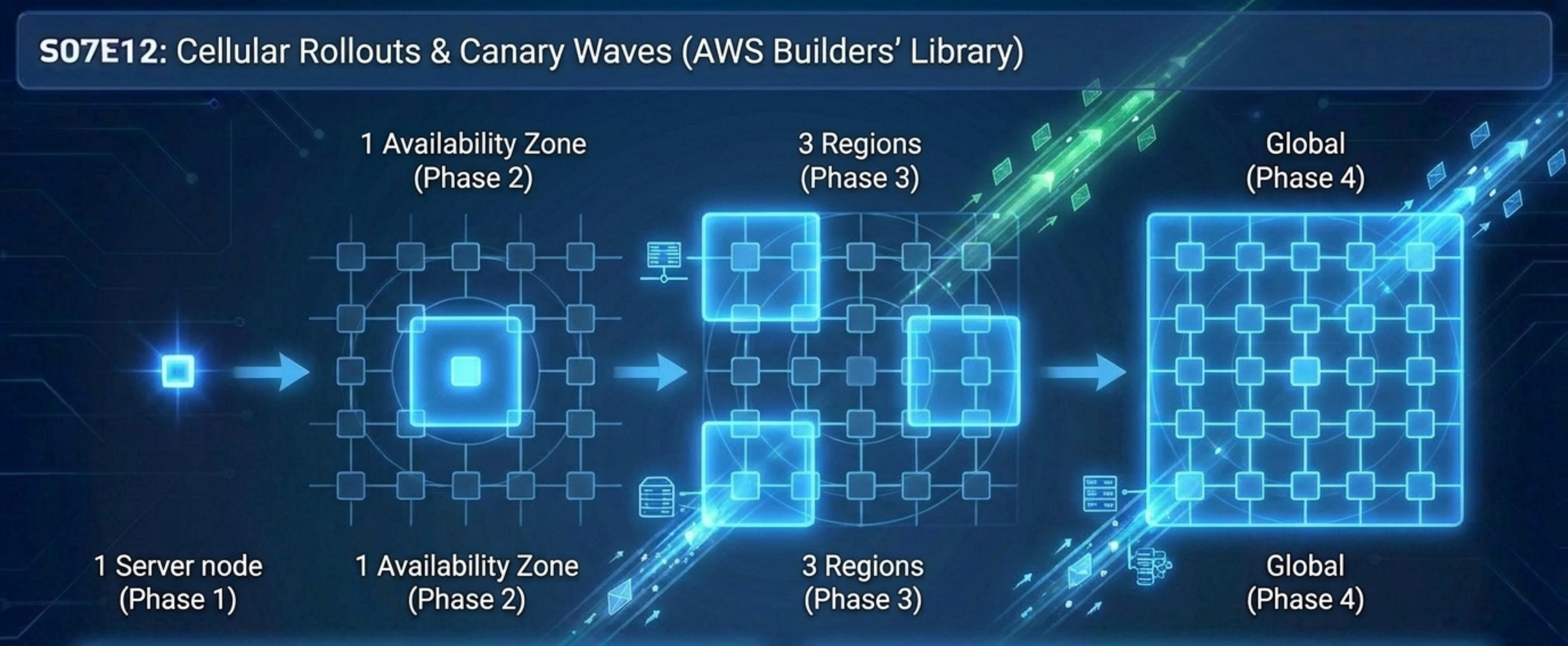
S07E12: Traffic Switching: Blue-Green Deployments



- **Deploy:** The Green environment receives the new code silently. Integration tests run in total isolation.
- **Switch:** The router redirects 100% of live user traffic to Green instantly.
- **Safety Mechanism:** If telemetry detects anomalies, an instant router flick sends traffic back to Blue.
- Provides rapid rollback without redeploying code.

Gemini Notebook

S07E12: Cellular Rollouts & Canary Waves (AWS Builders' Library)



Blast Radius Containment:

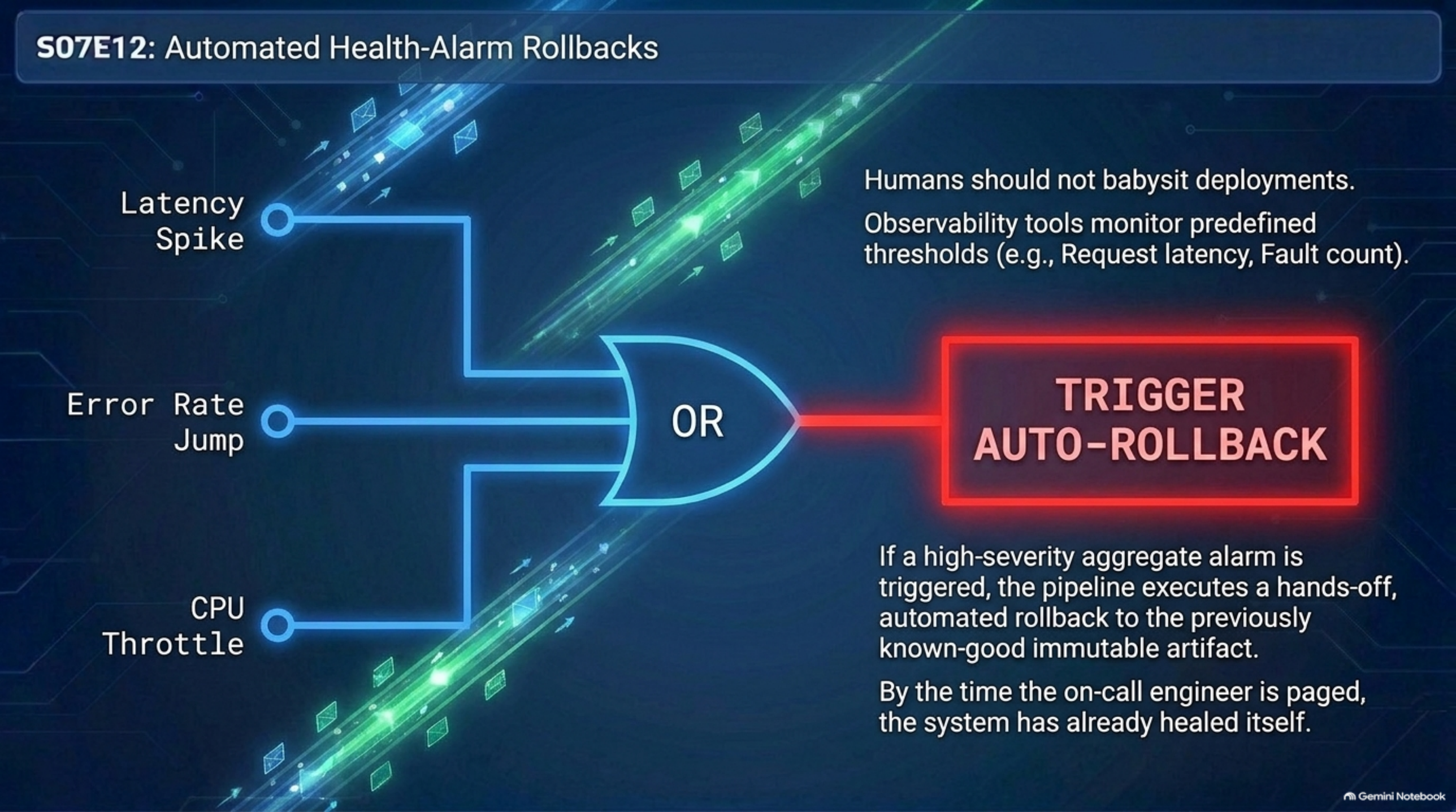
- The **One-Box Stage**: Deploy the artifact to a single 'canary' node first (serving max 10% of traffic).
- **Staggered Waves:** Expand deployment gradually across availability zones and regions.

Baking Windows:

- The automated pipeline pauses after each wave. It actively watches telemetry for slow-burning issues (e.g., memory leaks) before progressing to the next tier.

Gemini Notebook

S07E12: Automated Health-Alarm Rollbacks



Humans should not babysit deployments.

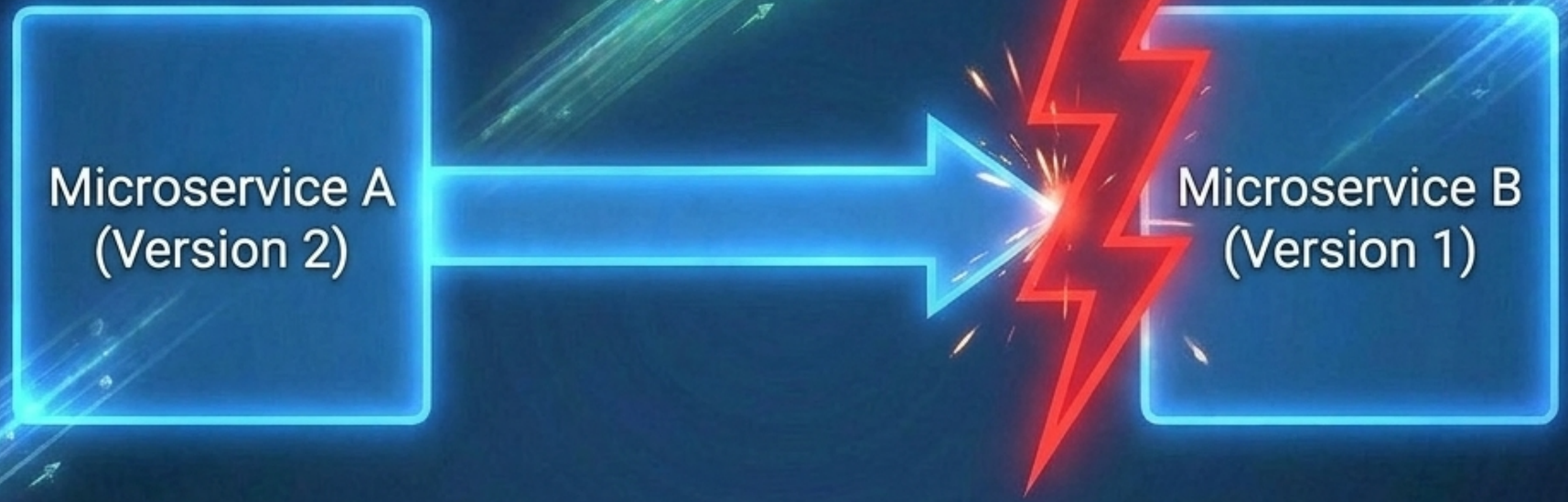
Observability tools monitor predefined thresholds (e.g., Request latency, Fault count).

If a high-severity aggregate alarm is triggered, the pipeline executes a hands-off, automated rollback to the previously known-good immutable artifact.

By the time the on-call engineer is paged, the system has already healed itself.

Gemini Notebook

S07E12: Version Skew & Rollback Safety



If an automated alarm triggers a rollback of Microservice A to Version 1, Microservice Microservice B must still be able to parse its requests.

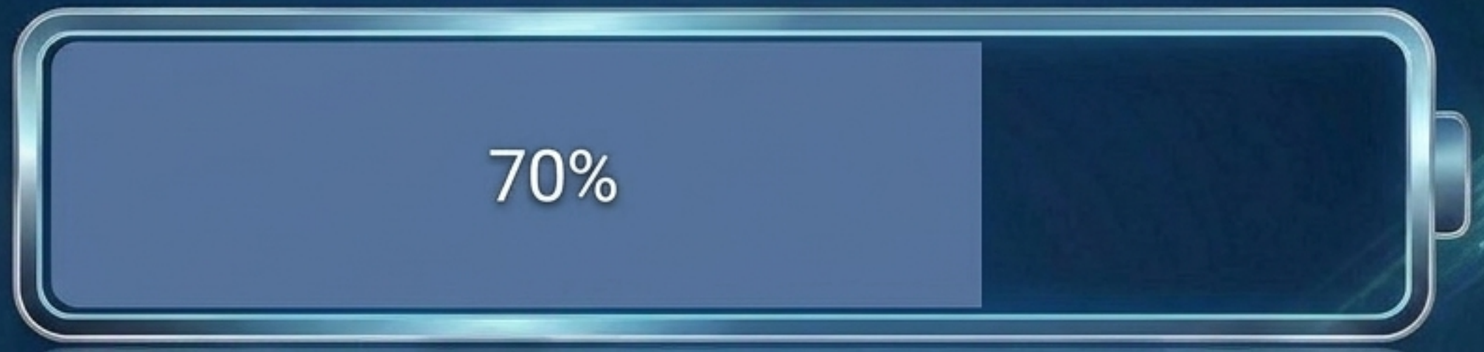
The Safety Rule: All code and schema changes must be backward-compatible.

Always deploy schema expansions before the code that relies on them.

Gemini Notebook

Google SRE Error Budgets

Error Budget



The Math:

If a service has a 99.9% uptime SLA, it has a 0.1% “Error Budget” for allowable downtime (approx. 43 minutes/month).

If Budget > 0 → Push code fast and take risks.

If Budget = 0 → All feature deployments freeze. 100% of engineering effort pivots to reliability, testing, and tech debt until the budget recovers.

Gemini Notebook

> STATUS: CI THEATRE

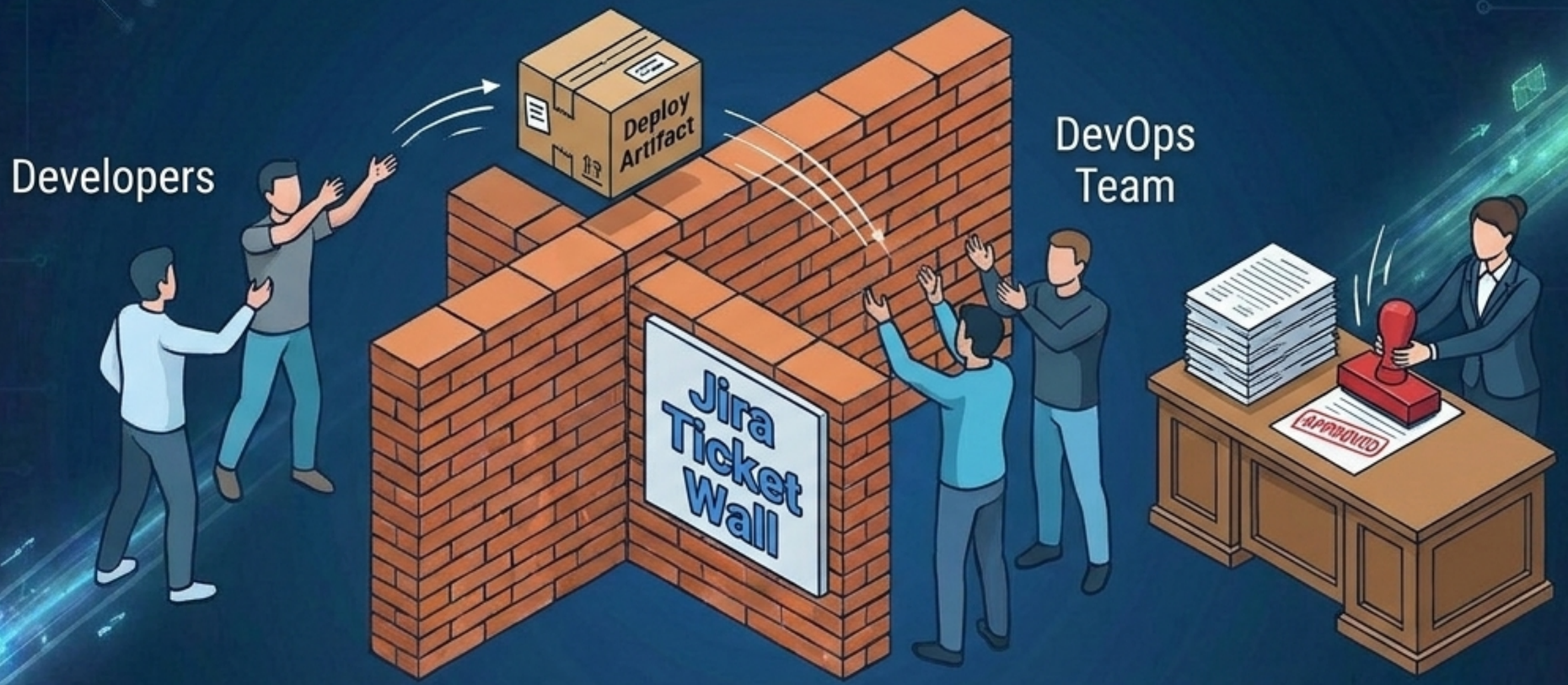
Anti-Pattern 1: CI Theatre



- Running automated Jenkins builds on isolated, long-lived feature branches is not Continuous Integration.
- It is just automated compiling.
- If developers aren’t integrating code into the shared mainline daily, the team is actively hiding integration conflicts and accumulating massive merge debt.

Gemini Notebook

Anti-Pattern 2: The DevOps Silo & CAB Theatre



- DevOps is a culture of shared responsibility, not a segregated job title to hand off work to.
- The CAB Illusion: Heavyweight, manual Change Approval Boards do not reduce risk; they increase batch sizes and rely on guesswork.
- Audit trails and compliance should be satisfied by automated pipeline controls and cryptographic artifact signing, not synchronous meetings.

Gemini Notebook

> STATUS: CI THEATRE

Anti-Pattern 3: Feature Flag Spaghetti



> ERROR:
$460M_KNIGHT_CAPITAL_GLITCH

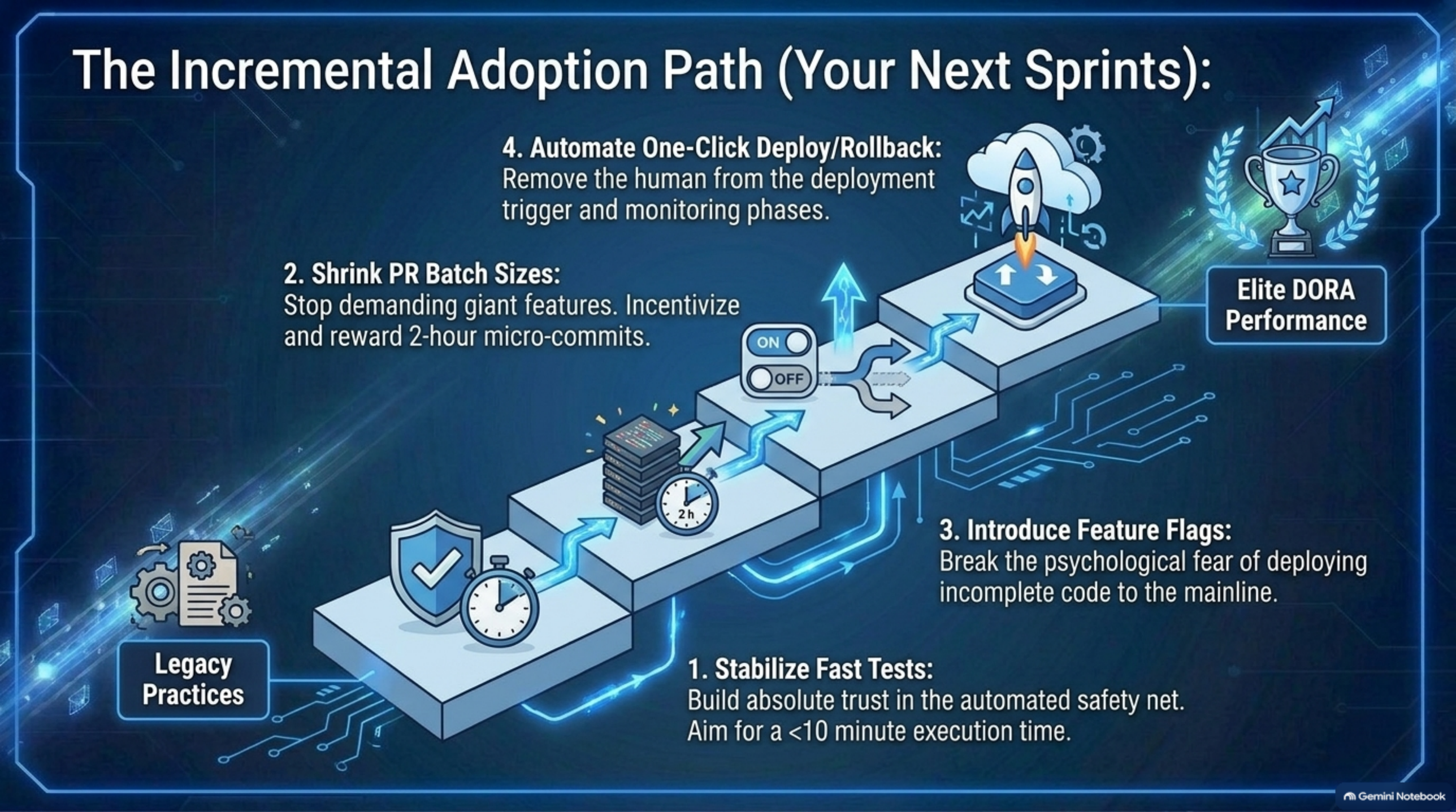
- Feature toggles are inventory with a carrying cost.
- Permanent, un-retired flag spaghetti creates an exponentially massive matrix of untested code path combinations.
- The Cure: Build automated expiration dates into toggles. Schedule toggle-removal tickets as a mandatory part of the 'Definition of Done'.

Gemini Notebook

The Incremental Adoption Path (Your Next Sprints):

4. Automate One-Click Deploy/Rollback:
Remove the human from the deployment trigger and monitoring phases.

2. Shrink PR Batch Sizes:
Stop demanding giant features. Incentivize and reward 2-hour micro-commits.



Elite DORA Performance

3. Introduce Feature Flags:
Break the psychological fear of deploying incomplete code to the mainline.

Legacy Practices

1. Stabilize Fast Tests:
Build absolute trust in the automated safety net.
Aim for a <10 minute execution time.

Gemini Notebook

> STATUS: TEASER CARD

Next Up — S07E05: Spec-Driven Development & AI Agents



- As cheap AI code generation puts unprecedented pressure on our repositories, human review bottlenecks will fatally break.
- The trunk-based CI/CD guardrails, test automation, and automated rollbacks we just built become the only things preventing catastrophic, high-speed automated failure.

> PREPARE YOUR PIPELINES.

Gemini Notebook