Xiaotime *Labs*

A position paper · Velocity argument

Everything as Code, including *governance*.

The velocity case for GRCDevSecOps. A companion paper for the CIO and CTO.

AWS named seven sub-disciplines of Everything as Code. None of them is governance. We think that is the next one.

Position paper.
For CIOs and CTOs whose engineering org ships faster than their GRC org can keep up.

VERSION
1.0
May 2026

The premise

> *If your engineering org ships software at the speed of code, your governance org has to operate at the speed of code or it becomes the tax on shipping.*

Everything as Code is about five years old. Born of Infrastructure as Code, configuration management, and GitOps. AWS's Well-Architected DevOps guidance names seven indicators. The principle is the same across all of them: take the artifact, put it in version control, run it through CI, automate its deployment. The speed at which you can change it goes from weeks to minutes.

Engineering organizations have absorbed this lesson. The CIO who runs an engineering org that does not version-control its infrastructure today is not a CIO. The category has moved from "leading-edge practice" to "table stakes" in fifteen years.

The governance stack has not absorbed the same lesson.

Policy still lives in PDFs. Controls still live in spreadsheets. Evidence still gets collected by humans. Audits still happen on quarterly cycles against point-in-time snapshots. The reporting layer that summarizes whether the controls are working still gets rebuilt by hand once a year.

For the CIO and CTO, this means the velocity gain at the engineering layer is taxed by the latency at the governance layer. Your developers ship in hours. Your compliance team responds in weeks. Your auditor visits twice a year. The cadence mismatch is the tax. The tax shows up as deals delayed at security review, releases held for compliance evidence, engineers re-paged to author audit responses, and quarterly board reports that lag the actual state of the system by a quarter.

The fix is not faster reporting. The fix is moving governance into the code layer.

The progression

Engineering's EaC journey followed a recognizable shape. Each category started as a manual practice, became a discipline with its own tooling, and ended up as table stakes.

DISCIPLINE	MANUAL ERA	TOOLING ERA	TABLE-STAKES ERA
Infrastructure	Hand-built servers	Terraform, CloudFormation	Pull request to provision

DISCIPLINE	MANUAL ERA	TOOLING ERA	TABLE-STAKES ERA
Configuration	Edited live	Ansible, Chef, Puppet	Versioned, reviewed, deployed
Network	Vendor console clicks	Network IaC, network policy code	Pull request to change a route
Documentation	Word docs in shared drives	Markdown, docs-as-code	Source of truth in repo
Image (compute)	Custom AMIs	Packer, image pipelines	CI builds the image
Data operations	Hand-tuned SQL	dbt, schema-as-code	Pull request to change a column

Each row took roughly five years to move from manual to table stakes. The category that was last in that list, data operations as code, landed as a discipline around 2019 and is now standard in any engineering org running on a modern data stack.

Governance is next. The pieces are already half-built: SOC 2 evidence in Vanta or Drata, control catalogs in AuditBoard or ServiceNow, security scanning in Snyk or Wiz, AI governance bolted on in spreadsheets. Each tool covers a slice. None of them is wired into the pull request, the CI pipeline, the merge gate, or the deploy event. They sit alongside the code, not inside it.

That is the gap. That is where the next round of velocity gain lives.

Four more categories of EaC

If AWS named seven, here are four more. They sit on the same maturity curve. They will follow the same trajectory.

We built the pipeline that puts them on it. The pipeline is two cooperating products that close the governance loop together. **AI-SDLC** is the operational arm: the pipeline runs, gates fire, verdicts get written, code ships. **ICRG** is the observer arm: every gate verdict, every merge, every deploy, every register update lands in a unified control register that projects to every framework the organization is bound by. Same activities, two roles in the same flow. Each category below names both.

1

Policy as code

A policy in a PDF is a policy you cannot enforce. A policy expressed as code that runs in CI is a policy that blocks the pull request when it is violated. The difference is whether the rule is a recommendation or a gate. Engineering teams have already accepted that bad code patterns get blocked at CI time: static analysis (SAST), dynamic analysis (DAST), container scanning, dependency scanning, migration smoke tests, type checking, and format checking are all standard CI gates today. Compliance policies belong on the same chokepoint.

AI-SDLC runs the gates: SAST, DAST, container scanning, dependency scanning, type checking, migration smoke tests, format checking, and an AI code review agent that reviews security, quality, patterns, tests, and dependencies before a human sees the diff. **ICRG** observes every verdict and updates control state. Same gates for a human engineer, an AI agent, a contractor, or Dependabot.

2

Controls as code

A control listed in an audit framework is a hypothesis. The control is real when telemetry can verify it. The control catalog becomes a queryable artifact, not a Word document. Each control is a function over telemetry that returns "pass" or "fail" with evidence attached. Audit becomes a query, not a project.

ICRG holds a unified register of controls across NIST CSF, NIST 800-53, HITRUST, and CSA AICM. **AI-SDLC** is the primary telemetry source: every PR, every gate verdict, every merge, every deploy lands in the register as a control-state observation. External tools add coverage on top.

3

Evidence as code

Evidence today is screenshots and CSV exports collected by humans against a deadline. Evidence as code means the evidence is generated by the system that runs the control, append-only and hash-chainable per org, time-stamped, and stored in a way that an auditor can query. The cost of producing a SOC 2 evidence package drops from weeks to a single query.

AI-SDLC writes evidence as a byproduct of shipping. Every PR, gate, merge, and deploy emits an event. **ICRG** appends each event to an append-only audit trail, cryptographically chainable per org, and renders framework-aware bundles for any audit lens from the same underlying register.

4

Audit as code

The audit itself runs against an event stream. The auditor opens the same query the system runs continuously. Findings, remediations, and posture deltas are visible as they happen. The audit visit becomes a review of a live dashboard, not a months-long evidence collection exercise.

AI-SDLC advances every change through a single chokepoint that writes the audit row inline. **ICRG** exposes the resulting event stream to the dashboard, the board view, and the auditor evidence pack, all reading the same source in real time.

THE THESIS, RESTATED

AWS says Everything as Code. We say Governance as Code. Your controls, your risk tolerance, your compliance posture, all versioned, testable, and auditable. Not documentation theater. Operational truth.

The category for the combination is GRCDevSecOps. We named it. We wrote about it. This paper is the velocity argument for the CIO and CTO who has to sign for the budget.

What it looks like to your engineering org

For the engineers, the immediate change is that fewer things stop their pull request from merging. Or, more precisely: the same number of things stop the pull request, but they stop it earlier, in CI,

with a clear error message and a documented fix, instead of stopping it weeks later when a compliance reviewer asks a question nobody on the team can answer.

A few concrete deltas:

- **Audit prep stops being a separate project.** The evidence already exists. The query against it is the prep.
- **Security reviews shorten.** A buyer's security team that wants a SOC 2 mapping gets a live link, not a six-week back-and-forth.
- **The auditor's questions are answerable in real time.** Continuous evidence means the auditor's call to the platform team gets answered on the call.
- **The compliance team stops sending questionnaires.** They subscribe to the event stream the engineering team already produces.

The hidden gain is in retention. Engineers stay where the work feels modern. An engineering org that does governance manually in 2026 is the same flavor of friction as an engineering org that ran SSH-into-prod deploys in 2018. It is not catastrophic. It is just slow, manual, and competitive teams have moved past it.

What it looks like to you

The CIO and CTO metrics your board asks about move:

- **Velocity, measured honestly.** Deploys per day rises when no gate is held by an off-pipeline review.
- **Time to audit-ready.** From quarters to zero. Continuous evidence means you are audit-ready every day, not the Monday after the auditor leaves.
- **Security review cycle time.** Enterprise deals close faster because the security review is a live link, not a document exchange that takes a fiscal quarter.
- **Compliance program cost.** Headcount in GRC stops scaling linearly with framework count. One register projects to NIST CSF, NIST 800-53, HITRUST, and CSA AICM, with SOC 2, ISO 27001, and NIST AI RMF supported as projection targets via the framework crosswalk.
- **Risk visibility.** The board view of control posture is live. You stop reporting last quarter's snapshot.

The board-facing consequence is harder to summarize because it is less a metric than a re-frame. The CIO who can say "every shipped feature carries its own governance trail, signed and queryable" sits in a different conversation than the CIO still organizing a quarterly evidence collection sprint to prepare for an external auditor.

The competitive consequence is harder to ignore. The buyers who care about how fast your security review runs are the ones writing the biggest contracts.

Objections we hear from the CIO and CTO seat

> ***This sounds like another transformation initiative.***
>
> It is not. The pipeline already exists. ICRG and AI-SDLC stand up against your GitHub and start emitting telemetry the register reads from on day one. The first sprint points the pipeline at one repo. External telemetry from tools you already run layers in afterward. Most organizations find three or four controls they thought needed a separate program were already observable in data the pipeline itself produces.

> ***Our auditors do not ask for this.***
>
> Not yet. They will. NIST AI RMF, the EU AI Act, ISO/IEC 42001, and the CSA AI Control Matrix all reference continuous monitoring as a first-class function. SOC 2's Trust Services Criteria already include continuous improvement. The auditors who adopt fastest will be the ones serving the largest enterprises. The asymmetry in audit readiness is going to widen before it narrows.

> ***Our compliance team will resist.***
>
> The compliance team that resists is the one whose role is being framed as "the people who collect evidence." When the role is framed as "the people who design the policy and review the operational evidence the system already produces," the resistance flips. The work moves up the value chain, not away.

> ***We have not fully implemented IaC. Why add four more categories?***
>
> You do not implement them. AI-SDLC ships the policy and pipeline layer. ICRG ships the controls, evidence, and audit layer. Adoption is connection, not construction. Teams that are mid-IaC adoption often see the governance categories light up faster than the infrastructure ones, because the pipeline that runs the policy gate runs before the infrastructure being governed is provisioned.

> ***This sounds like a vendor pitch.***
>
> It is. We built ICRG because the gap was real. We are also describing the gap honestly because if we are wrong about the category, no vendor wins, including us. Read this as an argument first.

The CIO/CTO move

Adoption is one thing: stand up the pipeline. ICRG and AI-SDLC are the pipeline. The CI gates, the control register, the audit trail, the evidence bundles all ship with the runtime. The pipeline itself is the primary telemetry source. Every PR, every merge, every policy decision, every deploy, every register update lands in the audit canon.

A pilot looks like:

1. **Stand up the pipeline against one repo.** One team, one quarter. AI-SDLC runs the gates. ICRG populates the register from pipeline telemetry. Same gates apply to a human engineer, an AI agent, a contractor, or Dependabot.
2. **Pick the framework lens.** NIST CSF, SOC 2, HITRUST, or CSA AICM. The register projects to all of them simultaneously. The lens is which one shows up on your dashboards and the auditor's evidence pack.

External telemetry from security tools you already run (Snyk, Wiz, Crowdstrike, dependency scanners, cloud posture tools) extends the register's coverage without changing the pipeline. Layer it in when the first quarter delivers signal.

What you measure: deploys per day, audit prep hours, security review cycle time. Before vs. during. Most teams have the first comparison within the quarter.

Why we wrote this

We built ICRG and AI-SDLC. Two cooperating runtimes that together close the loop attackers exploit.

ICRG owns the substrate. A unified control register across every framework that applies. Continuous evidence projected from live telemetry. A signed audit trail that updates the register on every change. Reporting surfaces for boards, auditors, and customer security reviews, all reading from the same event stream.

AI-SDLC owns the execution. Policy-aware gates that run on every pull request. An AI code review agent that evaluates the diff before a human sees it. Deploy automation that runs through the same merge chokepoint regardless of who or what opened the PR.

The two products are the implementation of the principles in this paper, extending the EaC discipline into policy, controls, evidence, and audit. We did not write a category paper because category papers were missing. We wrote it because we built the runtime and noticed the category needed a name.

The companion paper to this one is the GRCDevSecOps position paper. That paper makes the architecture argument for CISO and platform leadership. Read together, the two cover the velocity case and the architecture case for the same shift.

THE PRINCIPLE, RESTATED

We didn't write another standard. We made the best ones actually work. That holds at the governance layer just as it does at the engineering layer.

See the product: **xiaotimelabs.ai/icrg.html**

About Xiaotime Labs

Xiaotime Labs works with mid-market and enterprise teams whose engineering velocity has outpaced their governance velocity. Our focus is organizations that have invested in modern engineering practice and are now feeling the friction of governance practice that has not kept up.

Built by **Craig George** and **Stephan Hundley**. Two former CISOs and engineering operators who built ICRG because the gap was real and the existing tools were not built for the speed at which engineering ships today.

CONTACT
info@xiaotimelabs.ai

LEARN MORE
xiaotimelabs.ai/icrg.html

A technical companion covering architecture and implementation detail is available to prospective customers under NDA.

One more thing

If this paper reads like an argument that should be obvious in two years and is contentious today, that is the intent. The CIO/CTO seat is where this category gets adopted first, because that seat is the one most exposed to the velocity tax. The CISO seat will follow because the auditors will follow.

We built ICRG because we thought that way. You should evaluate whether you want to, too.