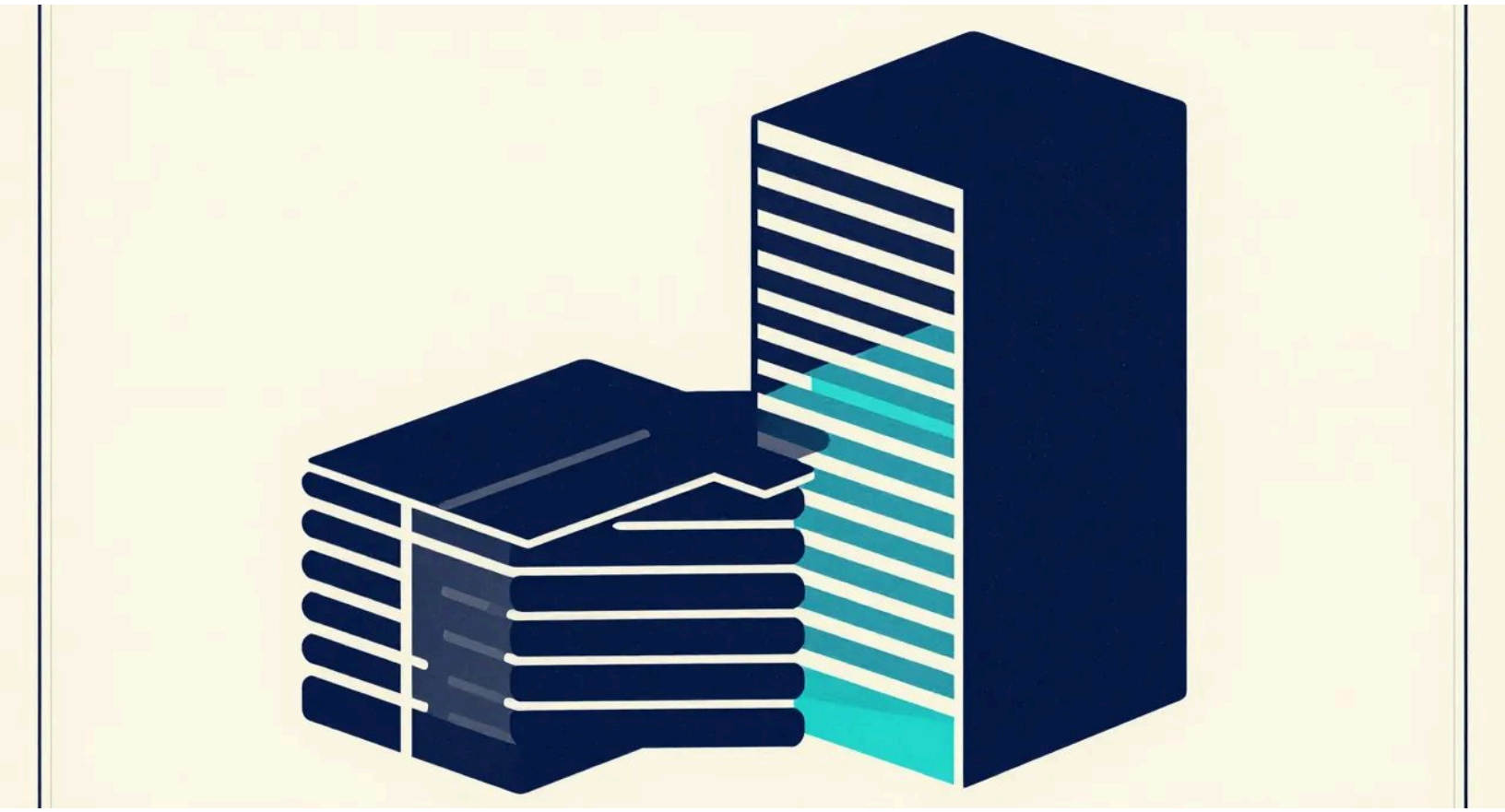


EXECUTIVE PAPER · INFRASTRUCTURE

Infrastructure as Code Is Business Continuity

Backups protect your data. Whether you can stand the systems around that data back up in hours or weeks depends on whether the environment is written down as code. Finance and technology leaders should fund it as continuity.

AUTHOR

Chris Watkins

SERVICE

Infrastructure

FOR

CTO, CFO · 13 min read

AT A GLANCE

The problem. Most companies back up their data and treat that as recovery. The servers, networks, permissions and settings that make the data usable were built by hand over years, and nobody can recreate them on demand. When a whole environment is lost, rebuilding it from memory takes days or weeks.

Why it persists. Hand-built environments work fine until they have to be rebuilt, and that day is rare. Configuration changes are made directly on live systems, so the record of what is running drifts away from reality. Infrastructure as code looks like an engineering preference, so it competes for budget with features.

The recommendation. Define every production environment in code held in version control, change it only through reviewed commits, and prove the definition works by rebuilding from it on a schedule. Report rebuild time and drift to leadership. Fund the work from the continuity and audit budget.

The Decision in Front of You

Every company that runs its own systems has a disaster recovery plan of some kind. Most of those plans assume the data can be restored. Far fewer can answer a harder question: if the environment that runs the data disappeared tonight, how long would it take to build a working copy, and who would know how?

Infrastructure as code (IaC) means describing servers, networks, firewall rules, identities and settings in text files that a tool reads to build the environment automatically. Those files live in a version-controlled repository, the same kind of system developers use for application code, where every change has an author, a reviewer and a timestamp.

The decision for the CTO and CFO is whether to fund that practice as a continuity control with its own budget line. This paper argues it should be. The return comes from faster recovery after a disaster, fewer failed changes in normal operations, and cheaper audits.

What's Happening

Data restores are only half of recovery

UniSuper, May 2024. UniSuper, an Australian pension fund with more than 647,000 members and over A$135 billion under management, lost its entire private cloud environment on Google Cloud [1]. Google's chief executive for cloud said an inadvertent misconfiguration during provisioning deleted UniSuper's subscription [1]. The fund had duplicated its setup across two regions, and the deletion

removed both [1]. Members could not reach their accounts for more than a week [2]. Recovery meant restoring hundreds of virtual machines, databases and applications, helped by backups UniSuper held with a separate provider [1].

The backups saved the data, and the week went into rebuilding the environment around it. Recovery from losing a whole environment depends on whether its machines, networks and permissions are defined somewhere outside the environment that was lost.

Delta Air Lines, July 2024. A faulty CrowdStrike security update crashed Windows machines worldwide on July 19, 2024. Delta's chief executive said the outage cost the airline as much as $500 million over five days and forced more than 5,000 cancellations [3]. Delta had to reset 40,000 servers manually, and the outage disrupted the crew-tracking system that matches pilots and flight attendants to flights [3]. The lesson for this paper is narrow. When every server is a unique, hand-maintained object, recovery speed is limited by how many people can touch machines in parallel.

Configuration changes are a leading cause of outages

Google Cloud, June 2025. On June 12, 2025, a policy change containing unintended blank fields was written to a database that replicates globally within seconds [4]. A code path added two weeks earlier had no error handling for blank fields and no feature flag, a switch that lets a change be turned on gradually [4]. Service Control, which checks policy and quota on Google Cloud API requests, crashed in every region. Most regions recovered in about three hours, and some services were impaired for about seven and a half hours [4]. Google's remediation commitments include propagating globally replicated configuration changes incrementally, with time to validate between steps [4].

Hand-built environments make this kind of failure more likely and harder to undo. NIST's guide to security-focused configuration management observes that many vulnerabilities trace back to misconfigured components [5]. A 2026 report by Reach Security, based on a survey of 250 U.S. security professionals and telemetry from more than 50 production environments, found 97% of organizations had a confirmed breach or near miss tied to security tool misconfiguration in the prior year [6]. Firewalls were the most common source [6]. The vendor sells configuration tooling, so treat the headline as an upper bound.

Delivery research ties stable change to recovery

Google's DORA research program surveys software delivery teams each year. Its 2024 report drew on more than 39,000 professionals [7]. In the 2024 performance clusters, elite teams had a change failure rate of about 5% and recovered from a failed deployment in under an hour. Low performers failed 40% of changes and took between a week and a month to recover [8]. The 2024 report also found teams with a dedicated platform team were about 6% more productive at the team level [9].

AI coding tools raise the stakes. DORA's 2024 report estimated that a 25% increase in AI adoption was associated with a 7.2% drop in delivery stability [10]. By 2025, 90% of respondents used AI in their work, and the association between AI adoption and instability persisted [11]. More changes, produced

faster, need a path to production that catches bad ones before they spread.

The standards already expect a recorded baseline

NIST SP 800-128 defines a baseline configuration as a reviewed, agreed set of specifications for a system that can change only through change control, and it recommends automated tools to detect when a system departs from its baseline [5]. The guide was published in 2011 and updated in 2019. The NIST Cybersecurity Framework 2.0, published in 2024, includes subcategory PR.PS-01, which asks that configuration management practices be established and applied. Its implementation examples include maintaining hardened baselines and monitoring for deviations from them [12]. ISO/IEC 27001:2022 added control A.8.9, configuration management, as a new control in that edition [13]. Auditors look for evidence that configurations are defined, documented, implemented, monitored and reviewed [13].

A repository of infrastructure code is that evidence. Each commit records who changed what and who approved it. A comparison between the repository and the live environment is a drift report.

SIDEBAR: WHAT "DRIFT" MEANS

Configuration drift is the gap between what an environment is supposed to be and what it is. It grows every time someone fixes a problem directly on a live server, opens a firewall port for a test and forgets to close it, or applies a patch to one machine and not its twin. NIST describes the goal as making sure configurations are not inadvertently altered from the approved baseline [5]. In a hand-built environment the approved baseline lives in people's memories and old tickets, so drift cannot be measured.

Why the Usual Approach Fails

Most disaster recovery programs protect data and assume the rest. Backups get tested by restoring a file or a database. Nobody tests building a new environment from nothing, because the only source is old wiki pages and the memory of whoever set it up.

Hand changes accumulate. An engineer resolving an incident at 2 a.m. changes a setting on the live system because that is fastest. The change works, the incident closes, and the setting now exists only on that server. After a few years, two servers that should be identical differ in dozens of ways. A replacement built from the original instructions behaves differently from the one it replaces.

Partial adoption creates false confidence. Many teams codify new cloud resources and hand-manage the older ones, or let engineers keep changing the live environment so the repository describes something that no longer exists. A definition never used for a rebuild is an untested backup.

The budget conversation also goes wrong. IaC gets pitched as developer productivity, where it competes with product features and usually loses. Its largest payoff, recovery from losing an environment, lands on a day nobody can schedule.

The Framework: The Rebuild Ladder

Score every production environment on a five-rung ladder. The rung is set by the weakest part of the environment, so a network defined in code does not lift a database server that was built by hand.

Exhibit 1. The Rebuild Ladder: only the top rung is proven continuity

Rung	Description	Can you rebuild it?	Typical rebuild time for a mid-size environment
0. Hand-built	Built by people, changed live, records partial	Only with the original builders	Weeks
1. Documented	Written build instructions, rarely tested	Slowly, with gaps found during the rebuild	One to two weeks
2. Scripted	Scripts exist, but live changes are still common	Partly; scripts lag the live system	Several days
3. Declared	Full definition in version control; changes only through reviewed commits	Yes, in principle	One to two days
4. Proven	Rebuilt from the repository on a schedule; drift detected and alerted	Yes, with a measured time	Hours

Source: author's framework. Rebuild times are the author's illustrative estimates and exclude data restore time, which is the same at every rung.

Four practices move an environment up the ladder.

Declare. Every component that production depends on, including networks, firewall rules, identities, DNS, certificates and server settings, is defined in code in one repository. Secrets stay in a vault, referenced by the code.

Gate. Changes reach production only through a reviewed commit and an automated pipeline. Emergency changes are allowed, and each one is committed back to the repository within one business day.

Detect. A scheduled job compares the live environment with the repository and alerts on any difference. Every drift alert is either committed to the repository or reversed.

Rebuild. At least twice a year, the team builds a complete copy of a production environment into an empty account or site from the repository alone, restores data into it, and records the time. The rebuild time becomes the recovery time objective the business can rely on.

What it is worth

The model below is illustrative for a hypothetical services company with about 120 production servers and 600 infrastructure changes a year. Replace each assumption with your own.

Exhibit 2. Illustrative annual value of moving from rung 0 to rung 4

Assumption or result	Hand-built (rung 0)	Proven IaC (rung 4)
Infrastructure changes per year	600	600
Change failure rate	15%	5%
Cost per failed change (engineering time plus business impact)	$5,000	$5,000
Annual cost of failed changes	$450,000	$150,000
Days to rebuild the environment after a total loss	10	1.5
Business cost per day of total outage	$150,000	$150,000
Cost of one total-loss event	$1,500,000	$225,000
Annual probability of a total-loss event	5%	5%
Expected annual cost of total loss	$75,000	$11,250
Engineering hours to assemble audit evidence	400	80
Annual audit preparation cost at $120 per hour	$48,000	$9,600
Total annual cost of these three risks	**$573,000**	**$170,850**

Source: author's illustrative estimate. Change failure rates are assumptions informed by the gap between DORA's 2024 performance clusters [8] and are not a measured effect of IaC alone. Rebuild time assumes 720 engineer-hours of hand work (6 hours per server) done by six engineers on 12-hour shifts, against 1.5 days to run the pipeline and validate. Data restore time is excluded from both columns.

On these assumptions the program removes about $402,000 a year of expected cost. If the first year costs $260,000 in engineering time, training and tooling, and later years cost $90,000 to maintain, the first year still returns about $142,000 and later years about $312,000. The change failure rate is the largest assumption. Halving the improvement, to 10% from 15%, still leaves about $252,000 of annual benefit.

Costs, Risks, and Tradeoffs

Exhibit 3. Tradeoffs of running infrastructure as code as a continuity control

Area	Cost or risk	Mitigation
Up-front effort	Codifying an existing environment takes months of senior engineering time	Start with the environment that carries the most revenue; import existing resources before rewriting them
Speed of urgent fixes	A reviewed pipeline is slower than logging in and changing a setting	Allow break-glass changes, and require a commit within one business day
Blast radius	Automation applies a bad change everywhere at once, as Google's June 2025 outage showed [4]	Roll changes out in stages with validation between them; require review on any change that touches every region or site
Skills	IaC tools and practices are new to many operations teams	Pair operations staff with experienced engineers; standardize on one toolset
Legacy systems	Some older platforms and appliances cannot be fully described in code	Codify everything around them; document and drill a manual rebuild for the remainder
When this is wrong	A small company running mostly on SaaS has little infrastructure to define	Keep vendor configuration exports and account settings under version control and stop there

Source: author's analysis.

IaC also concentrates risk in the pipeline and the repository. Whoever controls them controls production. Protect them like a payment system, with multi-person approval for changes to production, separate credentials, and an offline copy of the repository kept with the backups.

How to Act

Exhibit 4. A one-year roadmap

Window	Actions	Owner	Milestone
First 30 days	Score every production environment on the Rebuild Ladder. Pick the one with the highest revenue or regulatory exposure. Estimate its rebuild time honestly.	CTO with infrastructure lead	Ladder scores on file; one target environment chosen; current rebuild estimate reported to CFO
First 30 days	Add environment rebuild time to the continuity risk register alongside data recovery time.	CFO or risk lead	Rebuild time listed as a tracked risk metric
First 90 days	Codify the target environment, including network, identity and firewall rules. Route all its changes through reviewed commits and a pipeline. Start nightly drift detection.	Infrastructure lead with security	Target environment at rung 3; drift report delivered weekly
First 90 days	Store an offline copy of the repository with the backups. Require two-person approval for production changes.	Security lead	Repository recoverable without the primary source control system
First 365 days	Rebuild the target environment into an empty account or site twice and record the time. Extend the program to the next two environments. Map repository history to NIST CSF PR.PS-01 and ISO 27001 A.8.9 evidence.	CTO with compliance	Target environment at rung 4 with a measured rebuild under 48 hours; audit evidence drawn from the repository

Source: author's recommendation.

Questions to Ask Your Team

1. If our main production environment were deleted tonight, how long would it take to rebuild it, and when did we last test that number?
2. Which parts of production are defined in a repository, and which exist only on the live systems?
3. How many changes were made directly on production servers last month without a commit?
4. What would show us that a production server has drifted from its approved configuration?
5. If our source control system were also lost, where is the copy of our infrastructure code?
6. What share of our infrastructure changes caused an incident or rollback last quarter?
7. When the auditors ask for evidence of configuration control, what do we hand them, and how many hours does it take to assemble?

Closing

Define production in version-controlled code, change it only through reviewed commits, measure drift, and rebuild from the repository on a schedule. Then the recovery time in the continuity plan is a number the team has measured, and the audit evidence already exists in the repository's history.

Sources

[1] UniSuper and Google Cloud. "A joint statement from UniSuper CEO Peter Chun, and Google Cloud CEO, Thomas Kurian." May 2024. https://www.unisuper.com.au/about-us/media-centre/2024/a-joint-statement-from-unisuper-and-google-cloud.

[2] InfoQ. Coverage of the UniSuper Google Cloud outage. May 2024. https://www.infoq.com/news/2024/05/google-cloud-unisuper-outage/.

[3] NPR. "Delta's CEO says the CrowdStrike outage cost the airline $500 million in 5 days." July 31, 2024. https://www.npr.org/2024/07/31/nx-s1-5058652/delta-crowdstrike-outage-500-million-dollars.

[4] Google Cloud. Incident report for the June 12, 2025 Service Control outage. Google Cloud Service Health. 2025. https://status.cloud.google.com/incidents/ow5i3PPK96RduMcb1SsW.

[5] National Institute of Standards and Technology. "Guide for Security-Focused Configuration Management of Information Systems." NIST SP 800-128, includes updates as of October 10, 2019. 2011. https://scrm.aerospace.org/wp-content/uploads/2021/12/NIST.SP_.800-128.pdf.

[6] Channel Insider. "Configuration Drift Is a Growing Cybersecurity Risk: Report." Coverage of Reach Security, "Security Intent vs. Security Reality: Configuration Drift in the Age of AI." September 2026. https://www.channelinsider.com/security/configuration-drift-critical-cyber-risk/.

[7] InfoQ. Coverage of the 2024 DORA Accelerate State of DevOps Report. November 2024. https://www.infoq.com/news/2024/11/2024-dora-report/.

[8] Octopus Deploy. Summary of the 2024 DORA performance clusters. 2024. https://octopus.com/blog/2024-devops-performance-clusters.

[9] DORA. "2024 DORA Report Infographic." 2024. https://dora.dev/research/2024/2024-DORA-Report-Infographic.pdf.

[10] Google Cloud. "Announcing the 2024 DORA report." Google Cloud blog. 2024. https://cloud.google.com/blog/products/devops-sre/announcing-the-2024-dora-report.

[11] Stephens, R. Analysis of the 2025 DORA State of AI-assisted Software Development report. RedMonk. December 18, 2025. https://redmonk.com/rstephens/2025/12/18/dora2025/.

[12] CSF Tools. "PR.PS-01: Configuration management practices are established and applied." NIST Cybersecurity Framework 2.0 (NIST CSWP 29, 2024) reference. https://csf.tools/reference/nist-cybersecurity-framework/v2-0/pr/pr-ps/pr-ps-01/.

[13] Advisera. "ISO 27001:2022 Annex A control 8.9: Configuration management." https://advisera.com/iso27001/control-8-9-configuration-management/.

The views in this paper are the author's own and do not represent any employer. Scenarios are hypothetical. This paper is not legal, financial, or compliance advice.