A FIELD MANUAL

The Cloud Security Operating Model

Governance, runbooks and operational procedures for regulated financial institutions — including AI, agentic systems and what arrives next.

Bhavin G Patel

C|CISO · CISSP-ISSAP · CISM

cybersecleader.com

Thirteen runbooks · Eleven templates · Built to be adapted, not admired

Why I wrote this

I got tired of reading cloud security material that stops exactly where it starts being useful. It tells you that you need an exception process. It does not tell you what happens when the exception has no expiry date, or who is supposed to say no, or what you do about the four hundred that accumulated before you arrived.

So this is a working document. Take it apart. The runbooks in Part IV are written the way I'd want them written if I were the one being called at two in the morning: a named owner, an order of operations, and the failure mode that catches people. The templates in Part V are field lists you can paste into a spreadsheet this afternoon and start filling in badly, which is better than a perfect one you never build.

Where a decision genuinely turns on your firm's risk appetite, I say so instead of pretending there's one right answer. Appendix L collects the ones I don't think anybody can answer for you.

Who it's for

The person accountable when it goes wrong. A CISO, a head of cloud security, a technology risk lead, or the engineer who got handed the problem and a deadline. Part I and Part V read fine at executive level. Parts II through IV assume you're going to operate this thing, not just present it.

How to use it

Starting from nothing? Read Part I, then build RB-01, RB-02 and RB-03 in that order: account vending, guardrail change, exceptions. Those three carry most of the weight and everything else gets easier once they exist.

Already have a programme and an examination on the calendar? Start at Part II. That's where the evidence problem lives, and the evidence problem is what examinations actually turn on.

AI arrived before your governance did? That's the common case, and it's not a failure of your programme. Go to Part III and RB-08.

> **A NOTE ON CITATIONS**
>
> I've referred to regulations by name rather than paragraph number. Supervisory expectations move, and a document that cites a 2024 paragraph will quietly mislead you in 2027. Go to the source before you rely on any of this in a submission.

First published 2026 at cybersecleader.com. *The Cloud Security Operating Model: A Field Manual for Regulated Financial Institutions.*

Disclaimer

This manual is general information, not legal, regulatory or audit advice, and reading it does not create an advisory relationship between us. Regulatory obligations vary by jurisdiction, charter, size and business model, and they change. Nothing here substitutes for your own counsel, your own second line, or what your own supervisor has actually told you they expect. I accept no liability for decisions taken on the basis of this document. Verify before you rely.

Contents

PART ONE

The Model

1 · Why Operating Models Fail

A regulated institution almost never fails a cloud examination because it was missing a control. It fails because nobody could say who owned the control, who checked it, and how they knew it was working on the day the examiner picked at random.

The technology in cloud security is, by now, largely a solved problem. Every major provider ships encryption, logging, network isolation, identity federation and policy enforcement. The reference architectures are public. You can buy posture management from a dozen vendors.

And yet I have watched two firms with nearly identical architectures end up in completely different places. Different incident rates, different remediation speed, and very different experiences in the examination room. Same tooling, same reference design, roughly the same budget. One of them could answer the questions. The other one could not.

That variance is organisational, not technical. In my experience it comes down to four failure patterns, and I have seen all four more often than I would like.

Failure one: the central team becomes a queue

A firm centralises cloud security to ensure consistency. The central team is competent and quickly becomes the bottleneck for every environment request, firewall change and design review. Engineering teams, under delivery pressure, discover that the fastest path to production is the one that doesn't involve raising a ticket. Within eighteen months the firm has a rigorous control framework that governs a shrinking minority of its actual estate. Nobody lied; the controls simply stopped describing reality.

Failure two: the second line operates the tooling

The security function builds the guardrails, runs the posture console, triages the findings, and also attests that the control environment is effective. This isn't three lines of defence. It's one line wearing three lanyards. It feels efficient and it is the single most common structural finding in cloud examinations at mid-size institutions. An independent challenge function that built the thing it is challenging isn't independent.

Failure three: evidence is a project, not an output

Controls are implemented well. Then audit asks how a control operated in March, and a team spends three weeks assembling screenshots for a sample of thirty resources out of four thousand. Repeat quarterly, for each supervisor, each auditor, and each client due-diligence questionnaire. The firm isn't under-controlled. It is under-evidenced, and the cost falls on the same engineers who should be building.

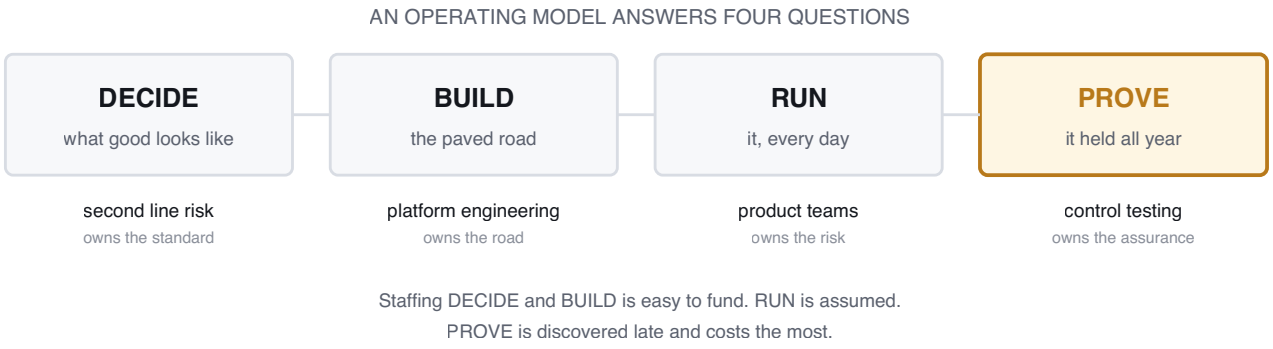
Failure four: exceptions become permanent

An exception is granted for a genuine reason with a note to review it annually. The annual review is a spreadsheet exercise conducted by someone who was not there for the original decision, and it renews. Five years later the firm has several hundred open exceptions, no one can explain most of them, and the control framework describes an aspiration rather than a state.

> **THE THROUGH-LINE**
>
> Every one of these is a failure of ownership rather than technology. Each is fixed by deciding who owns something, writing it down, and defending it — which is harder than buying a tool and is why it is so often skipped.

2 · The Four Questions

An operating model is not a control catalogue. It is the answer to four questions, each with a different owner. Most programmes answer two of them well, assume the third, and discover the fourth six weeks before an examination.



> The four questions, and the function that should own each. If two of these have the same owner, you have a structural problem before you have a technical one.

Decide — what good looks like

Somebody must define the standard: what encryption is required for which data classification, what network exposure is permitted, what identity model applies. This belongs to second-line technology risk, working from the firm's risk appetite. It is a small amount of writing that gets referenced constantly, and it must be specific enough to be testable. "Data must be appropriately protected" isn't a standard; it is a sentence that survived a committee.

Build — the paved road

Somebody must turn the standard into infrastructure that makes compliance the default. This is platform engineering, and it is a product function, not a policy function. Its output is landing zones, modules, pipelines and guardrails — the thing engineers actually consume.

Run — day-to-day operation

Product teams own the risk in their own workloads. They fix their own findings. This is the part most often assumed rather than designed: a firm implements guardrails and posture management, then discovers nobody agreed who remediates a finding in a team that has been reorganised twice since the workload was built.

Prove — it operated continuously

Somebody must be able to demonstrate, for any control, that it operated across the whole population for the whole period. This is control testing, and it is the question that determines your examination experience. Chapters 9 and 10 are entirely about it, because it is where most of the pain lives.

3 · Federation and the Three Lines

The useful question is never “centralised or federated”. It is which specific decisions sit where, written down, with names against them.

Centralised cloud security produces the queue described in Chapter 1. Fully devolved cloud security produces forty teams making forty different key management decisions. Every practitioner knows this, and the debate is usually conducted at a level of abstraction that resolves nothing. The way through is to enumerate the decisions and assign each one explicitly.

FIRST LINE — product and platform engineering

Owns the risk. Builds on the paved road. Runs the workload. Fixes its own findings.

Accountable for configuration, secrets and identity of its own services.

SECOND LINE — technology risk

Sets the standard. Challenges the design. Owns risk appetite and the exception register.

Does NOT build controls. Does NOT operate tooling. Does NOT approve its own work.

THIRD LINE — internal audit

Tests whether the first two lines did what they said. Independent reporting line.

Needs its own read-only cloud access. Borrowed evidence is not evidence.

> The three lines translate cleanly to cloud — provided the second line stops operating the tools it is meant to challenge.

The decision allocation

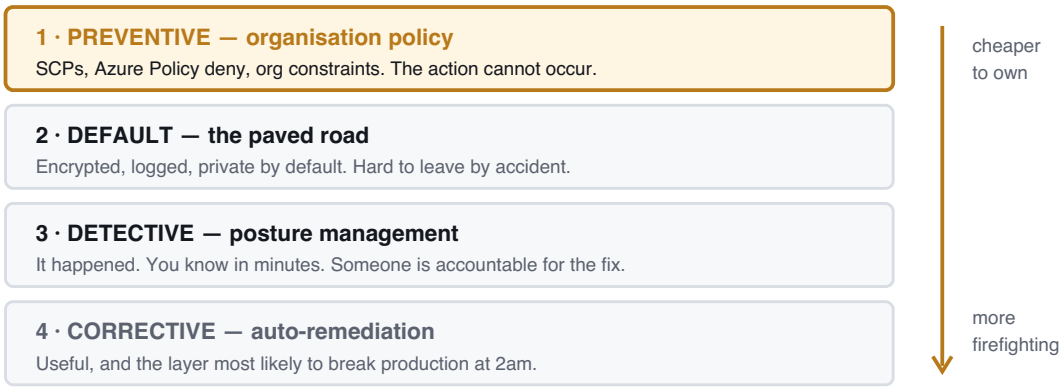
Decision	Owner	Consulted
Encryption standard by data classification	Second line	Platform, data office
Which preventive guardrails exist	Second line	Platform, first-line CISO
How a guardrail is implemented	Platform engineering	Second line
Landing zone architecture	Platform engineering	Second line, network
Whether a workload may go live	Product team	— (paved road pre-approves)
Whether an exception is granted	Second line	Business owner, audit informed
Remediation of a finding	Product team	Platform if road defect
Whether controls are effective	Internal audit	— (independent)
Risk acceptance above appetite	Named executive	Risk committee

THE MOST COMMON STRUCTURAL FINDING

If your second line operates the posture management console and triages its findings, it cannot credibly challenge the first line's control environment. Move operation of the tooling into platform engineering or a first-line security function, and leave the second line to set standards and challenge. This is unpopular internally and it is the correct answer.

4 · The Control Hierarchy

A control that makes something structurally impossible is worth enormously more than a control that tells you it happened. Most control catalogues flatten this into a list, which loses the entire insight.



> Every control pushed up this stack removes a category of incident rather than one instance of one.

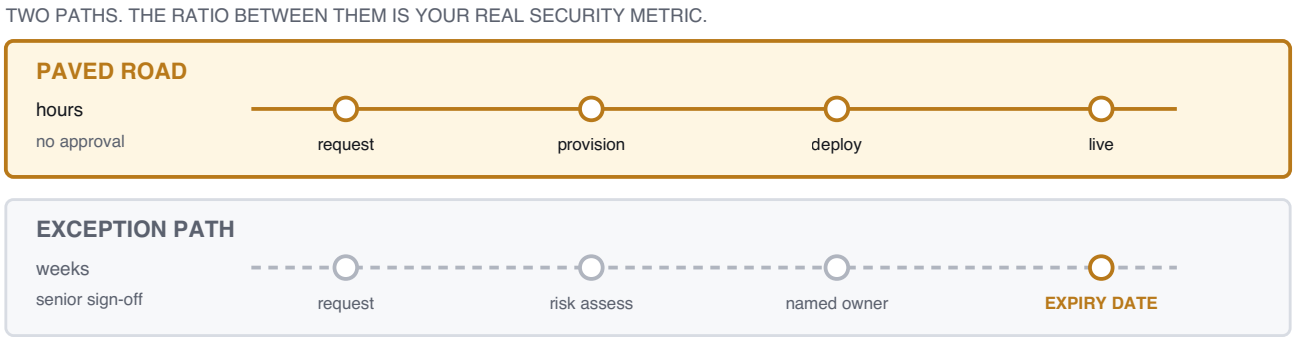
Rules versus defaults

The distinction between a rule and a default is political, not technical, and it decides adoption. A rule is something a team must comply with; compliance is a tax, and taxes are resented and eventually avoided. A default is something the team gets for free by doing nothing, and the tax falls instead on whoever wants to deviate. Identical control outcome, opposite relationship with engineering.

Practically: keep ten to fifteen genuinely non-negotiable preventive guardrails at the organisation level, with an exception path that runs somewhere uncomfortable. Everything else should be a default baked into the paved road. Appendix A gives a starting baseline.

5 · The Paved Road as a Product

How long does it take a product team to get a compliant, logged, encrypted, network-isolated environment with a working pipeline? If the answer is days, engineers will use it. If it is weeks and involves a form, your control environment is a story you tell.



If most teams are on the lower path, the upper path is not a road. It is a poster.

> The exception path should work, and should be visibly slower. The friction is the control.

What the road must include

An account or subscription with guardrails already attached. Network connectivity and egress control configured. Logging shipped to a central, immutable store the team can't alter. Identity federation with no static credentials. A deployment pipeline with branch protection and artefact signing. A secrets manager, wired up. Baseline monitoring and a cost budget. If a team has to assemble any of these themselves, that's where your variance and your findings will come from.

The two things that decide success

An owner accountable for adoption, not architecture. Someone whose performance is measured by the percentage of workloads on the road, who talks to teams that went around it, and who treats their reasons as defects in the road rather than failures of the driver.

Expiry dates on every exception. Not "review annually", an actual date on which approval lapses. RB-03 sets out the mechanics.

6 · Identity: The Real Perimeter

In a mature cloud estate, human logins are a small minority of authentications. The overwhelming majority are workloads authenticating to workloads — and most identity governance was built entirely around humans.

Joiners-movers-leavers, access recertification, privileged access management: all designed for people. None of them see service principals, managed identities, IAM roles, API keys held by SaaS integrations, or the credentials embedded in a pipeline. In most firms nobody owns the non-human population, nobody reviews it, and it grows monotonically because there's no leaver event.

Four positions the model must take

Area	Position
Workload identity	Services authenticate using short-lived federated credentials bound to the workload — OIDC federation, managed identities, IRSA. Static long-lived keys are a preventive guardrail violation, not a finding.
Standing privilege	Nobody holds production administrative rights permanently. Elevation is requested, justified, time-boxed and logged. The target for standing privileged accounts is zero, and the number you actually have is a metric worth reporting.
Break-glass	A independent path that works when the identity provider is down. Hardware-backed, stored under dual control, alarmed on use, and reviewed after *every* use rather than quarterly. See RB-05.
Key custody	A deliberate decision per data classification between provider-managed, customer-managed and hold-your-own-key. Chapter 7 covers the trade-off honestly.

Non-human identity governance

Treat it as a population to be managed, not a side effect. Every non-human identity needs an owner, a purpose, a permission scope, a credential rotation position, and a review. The practical starting point is enumeration: most firms can't currently produce the list. Producing it — and the count of identities with no owner — is usually enough to secure funding for the rest.

7 · Data, Residency and Keys

Every cloud control ultimately exists to protect data, yet data classification is the control most often inherited from a pre-cloud policy and never revisited.

Classification that drives automation

A classification scheme is only useful if it changes what infrastructure does. Four tiers is usually enough, and each tier should map mechanically to an encryption position, a permitted network exposure, a residency constraint, a retention period and a logging requirement. If your scheme has seven tiers and none of them change a Terraform variable, it's documentation rather than control.

Residency, sovereignty and the difference

Data residency — keeping data physically in a jurisdiction — is straightforward and every provider supports it with region constraints, which should be a preventive guardrail. **Operational sovereignty is the hard one:** guaranteeing which humans, in which jurisdictions, under which legal compulsion, can access systems and data. It materially constrains which services you can consume, and it is increasingly what supervisors and clients are actually asking about when they say "sovereignty".

The key custody trade-off, stated honestly

Model	What you gain	What it costs
Provider-managed	Simplicity, no availability risk you own, full service compatibility.	You cannot cryptographically exclude the provider. Adequate for most internal data.
Customer-managed (CMK)	Key lifecycle control, separation of duties, revocation, per-tenant isolation.	You now own rotation, availability and the consequences of deletion. The standard for regulated data.
Hold-your-own (HYOK)	Provider cannot decrypt without you. Genuine cryptographic exclusion.	You have created a hard availability dependency on your own HSM estate, and many managed services will not work. Rarely the right answer.

ON HYOK

HYOK sounds reassuring in a risk committee and creates an operational dependency most firms are not equipped to run. Before committing, ask who is on call for the HSM at 3am, what happens to every dependent service when it is unreachable, and whether the workload can even use the services you intended. If the answer to any of those is unclear, use customer-managed keys and spend the effort on key governance instead.

8 · Tool Consolidation: Where It Helps and Where It Hurts

Consolidation is pitched as an efficiency argument and bought as one. It is actually a risk argument, and it cuts both ways in a manner that deserves more honesty than it usually gets.

The short version: **consolidation trades many small independent risks for one large correlated one.** That's frequently the right trade. It is never a free one, and in a regulated institution it is a decision that belongs at a level where somebody is able to say no.

CONSOLIDATION REDUCES RISK

- Fewer integration seams to break
- One identity model, one permission story
- Consistent telemetry, so correlation works
- Fewer agents holding root on every host
- Less swivel-chair, fewer missed handoffs
- Cheaper to operate, so actually operated
- One evidence format instead of six
- Skills depth instead of shallow breadth

Most real incidents live in the seams between tools, not inside any one of them.

CONSOLIDATION INCREASES RISK

- One vendor compromise reaches everything
- That vendor is privileged on every host
- Monoculture: one blind spot is universal
- Detection from a single perspective only
- Exit becomes practically infeasible
- Negotiating leverage gone at renewal
- Becomes a critical third party under DORA
- Their outage is your blind window

A security platform is the most privileged thing in the environment.

> Both columns are true at once. The question is never whether to consolidate, but which specific things must stay separate.

The case for is stronger than sceptics allow

Most real incidents aren't missed because a tool failed to detect something. They are missed in the seams — one product's alert never reached the other product's queue, or two half-signals sat in different consoles and nobody joined them. Fewer seams means fewer of those failures. A platform one team knows deeply outperforms six that nobody has time to tune, and a tool that's cheap to operate is a tool that actually gets operated rather than admired in a procurement deck.

The case against is not about features

It is that a consolidated security platform is the most privileged thing in your environment: an agent on every host, read access to everything, frequently the ability to execute. Consolidating that concentrates precisely the capability an attacker most wants, and supply-chain compromise of a security vendor isn't hypothetical. The blast radius scales exactly with how successfully you consolidated.

Alongside it sits monoculture. When one engine makes every detection decision, its blind spot is your blind spot everywhere, with no second perspective to notice. Diversity in detection is valuable and genuinely expensive, and pretending otherwise in either direction is how firms end up surprised.

Three things to keep deliberately separate

Keep separate	Why
The evidence store	If the product that enforces controls also holds the only record of whether they operated, you have no independent record. Write control test results to storage the tooling cannot modify.
Break-glass	The recovery path must not share a failure domain with the thing it recovers from. Break-glass cannot depend on the consolidated identity platform.
One independent detection signal	Not a second full platform — that undoes the benefit. A narrow, differently-sourced signal on your highest-value assets that would notice if the primary went quiet. Cloud-native provider logging is the cheapest version and comes from a different vendor by definition.

THE BOARD-LEVEL FRAMING

Past a certain point a consolidated security platform becomes a critical third party in its own right, carrying the concentration assessment, exit plan and resilience testing that designation implies. Firms consolidate for operational simplicity and then discover they have built exactly the dependency their supervisor asks about. Make the trade deliberately, record it as a risk decision, and revisit it at renewal, which is also the moment you discover how much leverage you gave away.

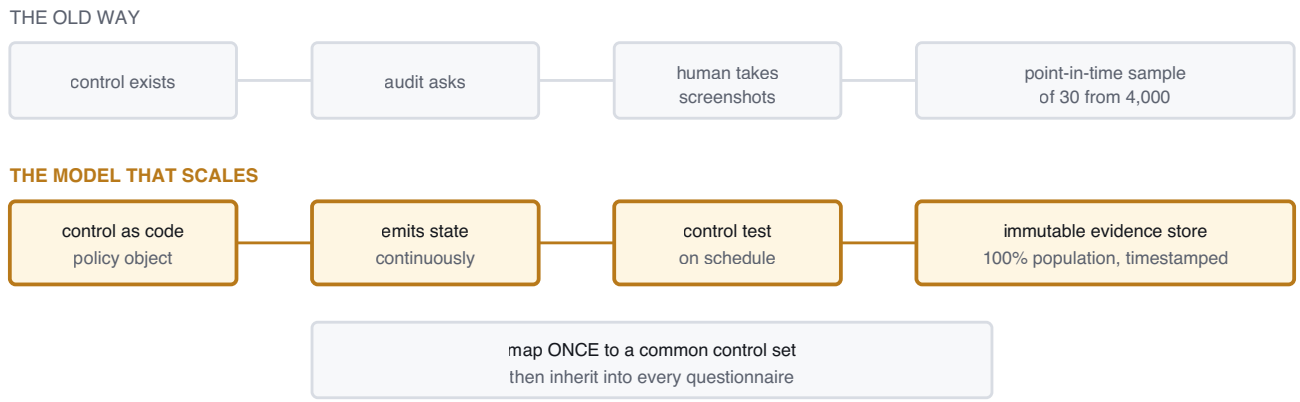
PART TWO

Proving It

9 · Evidence as a By-Product

If producing evidence is something humans do, it will be done late, inconsistently, and for a sample. If evidence is emitted by the control itself, it's continuous, complete and boring, which is exactly what you want it to be.

This chapter is the difference between a cloud programme that survives supervision comfortably and one that surrenders a quarter of its engineering capacity every audit cycle. It is also the least glamorous work in the book, which is why it is usually deferred until the first examination makes it urgent.



> Sampling is a legacy of manual evidence. When the control emits its own state you can test the entire population, which is both cheaper and a stronger assertion.

What a control test actually is

A control test is a scheduled job that answers one question about the whole population and writes the answer somewhere immutable. "Every storage bucket in every account had encryption enabled at 03:00 today — 4,182 of 4,182 pass." It runs daily. It's versioned in source control, and its output accumulates into a record that spans the period.

Three properties matter. It must cover the **whole population**, because a sample invites the follow-up question about the rest. It must be **timestamped and immutable**, written to a store the tested teams can't alter, object storage with versioning and object lock is sufficient and cheap. And it must record **failures as well as passes**, because a test that has never failed is usually a broken test rather than a perfect environment.

Evidence for things that are not resource states

Some controls aren't queryable configuration — approvals, reviews, incident handling. Those still need evidence, and the answer is the same: make the system of record produce it. If exception approvals live in a ticketing system with a defined workflow, the export is your evidence. If they live in email, you have no control. Design the workflow so its normal operation leaves a record.

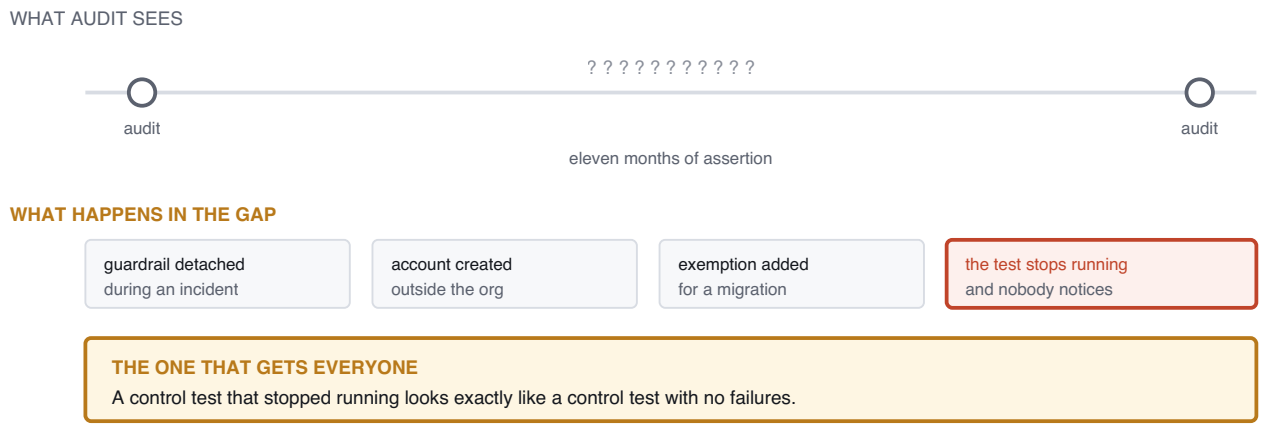
Retention

Match evidence retention to your longest examination look-back plus a margin — typically seven years for a regulated institution, but confirm against your own obligations. This is cheap in object storage and expensive to reconstruct.

10 · Proving It Between Audits

Ask a room of cloud security leaders whether their controls are effective and every hand goes up. Ask whether they can prove it for the fourteenth of last month and the room goes quiet.

That gap is the most useful question you can ask about your own programme, because it separates two things people routinely conflate: an assertion about the present, and evidence about a period. Point-in-time testing means asserting a state you last verified up to eleven months ago. Everything that matters happens in between.



> Continuous assurance is not more testing. It is testing that fails loudly when it stops.

The silent failure mode

A control test that has stopped running produces exactly the same dashboard as a control test that's passing. Green means "no failures reported" — and no failures reported is also what you get from a broken scheduler, an expired credential, a permissions change, or a query returning an empty population because somebody renamed a tag.

If you have no alert for the *absence* of results, your assurance has a silent failure mode, and it will fail silently at exactly the moment something changed. This is the cloud-assurance equivalent of a smoke detector with a dead battery: indistinguishable from a working one until it matters.

What continuous assurance actually requires

Requirement	What it means in practice
Scheduled execution with recorded results	Tests run on a cadence and write output whether they pass or fail, so the record is continuous rather than reconstructed on demand.
A dead-man's switch per test	Alert when a test has not reported within its expected window, and treat that as a **control failure**, not a monitoring nuisance. This is the single highest-value addition most programmes are missing.
Coverage as a measured number	What percentage of the estate is under test. A test covering 60% of accounts is a 40% blind spot nobody has quantified.
Boundary drift detection	Something watching for new accounts, detached policies and changed organisational structure, those changes make every other test irrelevant for the resources they touch.
Independence from the enforcing tool	Per Chapter 8: the record of whether a control worked must not live only inside the product that enforces it.

THE HONEST SELF-ASSESSMENT

If an examiner asked you today to evidence that a specific control operated on a specific date four months ago, would you run a query or start a project? Most firms would start a project, and most firms believe they would run a query. The distance between those two answers is your actual assurance position. RB-13 is the review that closes it.

11 · The Control Library and Framework Mapping

A large institution answers to several supervisors, an external auditor, PCI, and a queue of client due-diligence questionnaires — all asking the same questions in different vocabulary.

Firms that map controls once to a common framework and inherit outward spend a fraction of the effort of firms that treat each request as a fresh project. The Cyber Risk Institute Profile exists precisely because the sector became tired of this; NIST SP 800-53 and the NIST Cybersecurity Framework serve the same purpose.

Structure of a control library entry

Field	Purpose
Control ID	Your own stable identifier. Never a regulator's, those change.
Statement	One testable sentence. If you cannot write a query for it, rewrite it.
Owner	A named role, not a team mailbox.
Type	Preventive / detective / corrective, and automated / manual.
Implementation	Where it lives — SCP name, policy ID, pipeline stage.
Test	The scheduled job that proves it, and its frequency.
Evidence location	Where output lands. Must be specific.
Framework refs	CRI, 800-53, CIS, PCI, internal policy. Many-to-many.
Exceptions	Link to open exceptions against this control.

Appendix E gives worked example rows. The discipline that matters: every control has exactly one owner and at least one automated test, or it is explicitly recorded as manual with a justification. The count of controls with no automated test is one of your six board metrics, and it should be visibly falling.

12 · Metrics That Change Decisions

Most cloud security dashboards measure the wrong thing beautifully. Number of findings is not a measure of anything except how hard you are looking.

The test for a metric is simple: if it moved by 30% in either direction, would anyone do anything differently? If not. It's reporting, not measurement. Six numbers pass that test.

SIX NUMBERS FOR THE BOARD PACK

PAVED ROAD ADOPTION
% of new workloads with no exception. Falling = nothing else matters.

TIME TO COMPLIANT ENVIRONMENT
Decides whether engineers cooperate or route around you.

EXCEPTIONS PAST EXPIRY
The most honest number you have. Should be near zero.

MTTR BY SEVERITY
Split auto-remediated vs human. The gap is your scaling problem.

CONTROLS WITH NO AUTOMATED TEST
Your evidence debt, stated as a number.

UNREGISTERED AI & NON-HUMAN IDENTITIES
Shadow adoption. Zero means you are not looking.

Absent by design: total findings, tool coverage, training completion, policy count.
Those measure activity. None of them tell you whether the operating model works.

> Appendix F expands these into a board pack format with commentary prompts.

PART THREE

AI and What Comes Next

13 · AI as a New Control Boundary

Generative AI is not simply a riskier workload. It violates three assumptions your existing control set was built on, and that is why bolting it onto an existing framework does not work.

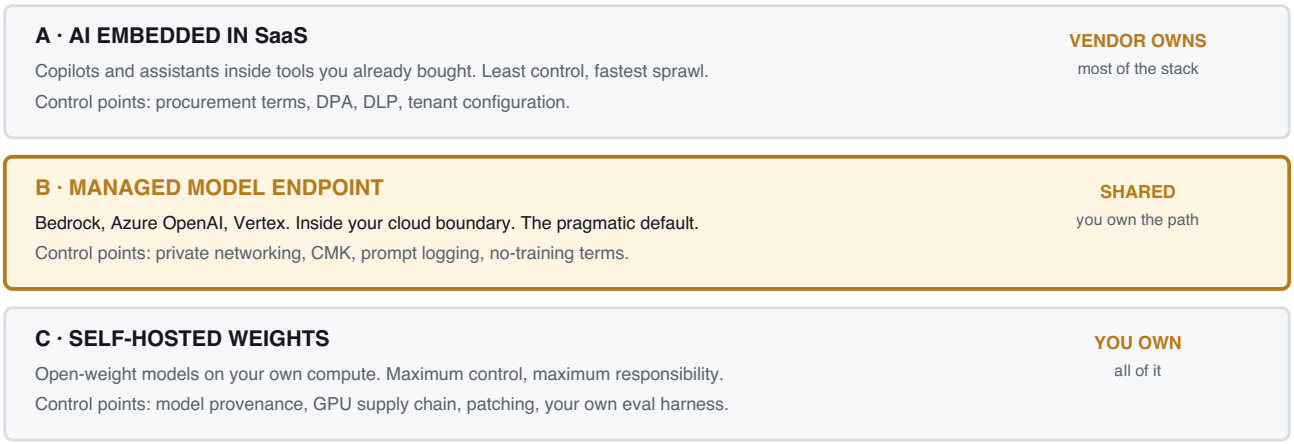
The three broken assumptions

The data leaves your inspectable boundary. Your controls assume you can see where data rests and who touched it. A prompt carries customer data into a service whose internals you can't inspect. No posture management tool flags this, because nothing is misconfigured — the system is working exactly as designed.

The component is non-deterministic. Every control test you own asks a binary question: is encryption enabled, is the port closed. "Does this model behave acceptably" has no binary answer, only a distribution over inputs. Your control testing function has never had to reason about that, and neither has your auditor.

In agentic form, it acts. A model that generates text is a data-handling problem. A model wired to tools is an identity with permissions — and unlike every other actor you govern, its instructions arrive from content it reads. Chapter 14 covers this separately because it is the highest-consequence part.

Three deployment patterns



> Most regulated firms should standardise on B and govern A ruthlessly. C is for a small number of genuinely justified cases — usually data that cannot leave, or economics at very high volume.

Pattern A is what catches firms out. Nobody runs a governance programme to adopt it; a vendor ships an AI feature into a product you already own, and a model begins processing customer data under terms nobody re-read. Your first control here isn't technical. It is a procurement standard stating that no product ships to production with AI features enabled until the data processing position is documented, plus a tenant configuration default of off.

The new control surface

Model and system inventory

You can't govern what you can't enumerate. This is the SaaS sprawl lesson, relearned. Appendix C gives the schema; RB-08 gives the intake process that populates it.

Prompt and completion logging, with a decision attached

You need it for investigation and to demonstrate control. It also means you're now storing a corpus of customer data in a new place with its own retention, access and discovery obligations. Log it — and classify the log store at the sensitivity of the most sensitive thing that can appear in a prompt, which is almost always higher than people assume.

The retrieval boundary

This is the most under-appreciated risk in enterprise AI. Retrieval-augmented generation indexes your documents so the model can answer from them, and the index typically inherits the permissions of whoever built it rather than of the person asking. That's a textbook confused deputy: a junior analyst asks a question and receives an answer synthesised from the board pack.

> **NON-NEGOTIABLE**
>
> Authorisation must be enforced at retrieval time, per requesting user. A design that retrieves broadly and filters afterwards, or that relies on the model declining to answer, is broken. Test this explicitly with a low-privilege account before go-live. It's step 7 of RB-08.

Evaluation harness

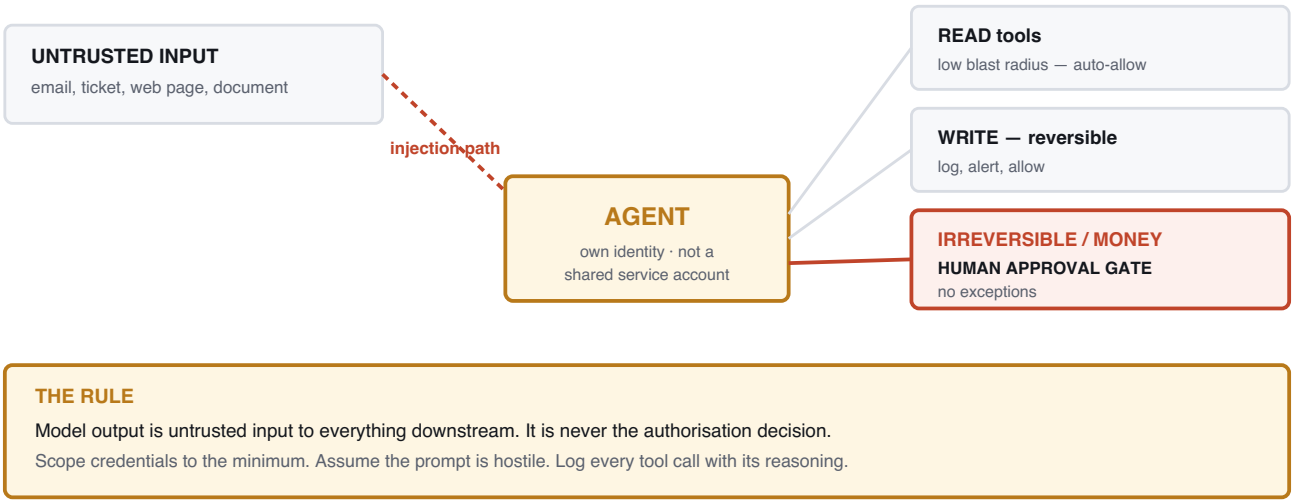
Regression testing for behaviour rather than correctness. It answers the model-risk question about ongoing monitoring, and it is the only way you will notice that a provider's model update has changed how your system behaves. Providers deprecate and update models on their own schedule; without an eval harness that's a silent change to a production system.

Cost and rate limits as an availability control

A runaway agent is a denial-of-service attack on your own budget. Token spend limits belong in production monitoring with alerting, not in a monthly finance report.

14 · Governing Agentic AI

When a model can call tools — query a database, move money, open a ticket, deploy code — it becomes an actor in your environment whose instructions can arrive from content it reads. Prompt injection stops being a content-filtering nuisance and becomes privilege escalation.



> Classify every tool by blast radius and reversibility, then gate accordingly. This is the highest-value AI control available to you.

Six design principles

1. **Its own identity.** Never a shared service account. Actions must be attributable to a specific agent instance.
2. **Minimum scope.** An agent that summarises tickets needs no write access to anything. Derive permissions from the tool list, not from convenience.
3. **Tools classified by blast radius.** Read, reversible-write, and irreversible. The classification is the control.
4. **A permanent human gate on irreversible actions.** Money movement, production deployment, data deletion, external communication. Not a temporary measure during pilot.
5. **Every tool call logged with its reasoning.** You will need this for incident reconstruction, and there is no other way to answer “why did it do that”.
6. **Deterministic authorisation.** The model may *request* an action; something deterministic decides whether it is permitted. If the model’s output is what grants access, prompt injection is a complete bypass.

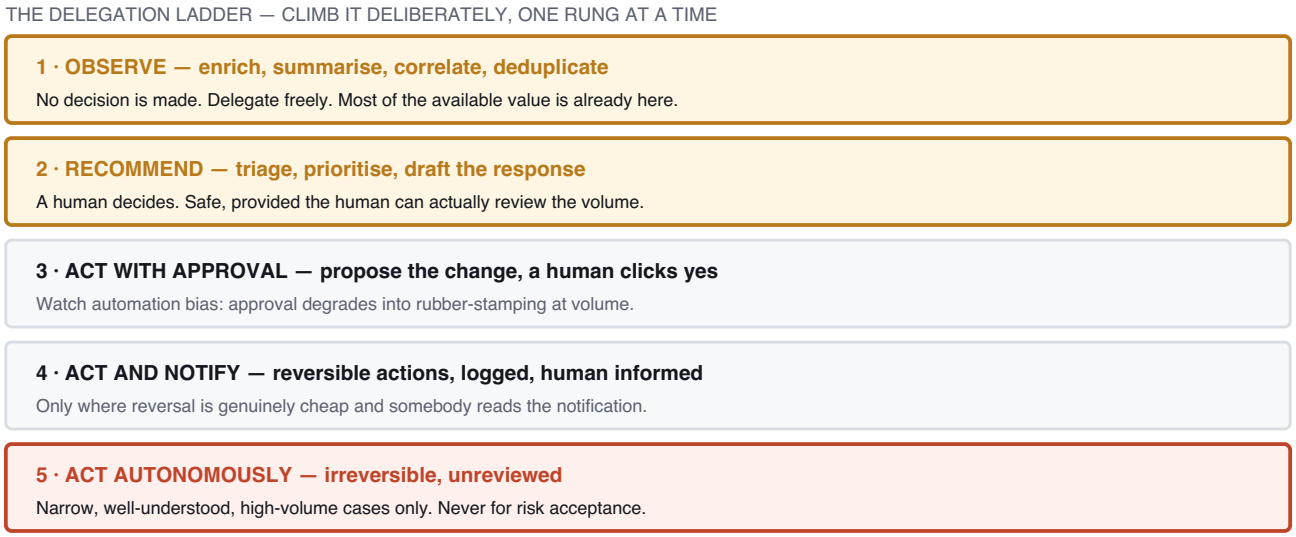
Agents as a workforce

If agents perform work people used to, they need the governance people have: an inventory, an owner, an access review, and a leaver process for when the agent is retired. Most firms have no leaver process for non-human identities at all, which is how orphaned credentials accumulate. Chapter 6 and RB-09 connect here.

15 · Which Security Decisions Can You Hand to AI?

Every security leader is about to be asked this by their board, and “we are evaluating it” will stop being an acceptable answer quite soon. Both instinctive postures are wrong.

Refusing all delegation means drowning in volume you can’t staff. Delegating by enthusiasm means discovering the limits during an incident. The useful move is to stop asking “can AI do this?” and start asking **“what happens when it does this wrong, and would we find out?”** That converts a capability question into a risk question — and risk questions you already know how to answer.



> Most institutions should operate confidently at rungs 1 and 2, cautiously at 3, and treat 4 as a case-by-case decision with a named owner.

The four tests

Test	What it catches
Is it reversible?	Blocking an IP for an hour is reversible. Deleting a resource, sending an external communication, or closing a customer’s account is not. The hardest line and the easiest to defend.
Is being wrong detectable?	The under-appreciated one. If AI misprioritises an alert you find out when the incident matures — too late, and you never learn how many others it got wrong. Decisions whose errors are *unfalsifiable* are the most dangerous to delegate, however reversible they look.
What is the blast radius?	The same decision at ten instances a day and at ten thousand is not the same decision. Autonomy multiplies correctness and error equally.
Can a human review it at this volume?	If the system produces four hundred recommendations a day and one analyst approves them, you do not have oversight — you have a person clicking yes.

Where the line sits today

Comfortable to delegate: alert enrichment and correlation, deduplication, log summarisation, first-pass triage and severity suggestion, drafting incident timelines, mapping a control to framework references, first drafts of policies and questionnaire answers, and identifying which of four thousand findings share one underlying defect. High volume, low individual consequence, human sees the aggregate.

Not delegable, and unlikely to become so: accepting risk. Approving an exception. Signing a regulatory attestation. Deciding a control is effective. Determining whether an incident is notifiable. Granting privileged access.

These aren't hard because models are insufficiently capable. They are hard because **accountability is personal and non-transferable.** A named individual owns the decision under your governance framework and, in several jurisdictions, under a senior manager regime carrying personal consequences. You can't delegate an accountability you're not permitted to give away, and no advance in capability changes that.

> **THE FAILURE MODE NOBODY PLANS FOR**
>
> Automation bias. Give a competent analyst a system that is right 95% of the time and within a month they stop evaluating and start confirming. Your rung-3 control quietly becomes rung 5 without anyone deciding to move it. If you delegate at rung 3 or above, **measure the human override rate** — and if humans are overriding almost nothing, your oversight has already stopped working. Appendix J is the matrix for recording these decisions.

16 · AI Under Financial Regulation

Banks have an advantage most sectors lack: you already have a model risk management framework. The instinct to treat AI governance as greenfield is usually wrong.

Extend model risk management, do not replace it

SR 11-7 has governed model development, validation and ongoing monitoring in US banking for over a decade, and equivalent expectations exist elsewhere. Generative AI systems are models. Start there.

The strain point is real: SR 11-7 assumes you can validate a model — understand its inputs, test its logic, document its limitations. A foundation model you did not train, can't inspect, and which the provider updates on their own schedule doesn't satisfy that assumption. The resolution most firms are converging on is a distinct **third-party model** category in which validation shifts from inspecting the model to testing the system around it: the evaluation harness, the guardrails, the human review step, and the fallback behaviour when the model is unavailable or wrong.

What else applies

Regime	What it means for an AI system
EU AI Act	Obligations by risk tier. Several core banking uses — creditworthiness assessment among them — fall in the high-risk category, attracting documentation, human oversight, logging and monitoring duties. Check current timelines; they phase in.
Fair lending / ECOA	Obligations do not soften because the model is novel. If a system contributes to a credit decision you still owe a specific adverse action reason, which constrains how opaque a component can sit in that path.
DORA	Your model provider is an ICT third party. That pulls AI into the register of information, concentration risk assessment and exit planning.
Privacy (GDPR et al.)	Prompts and completions are personal data when they contain it. Lawful basis, retention, subject access and deletion all apply to your prompt log.
Records retention	If an AI system participates in a customer interaction or a decision, its inputs and outputs may be records with retention obligations. Decide deliberately rather than discovering it later.

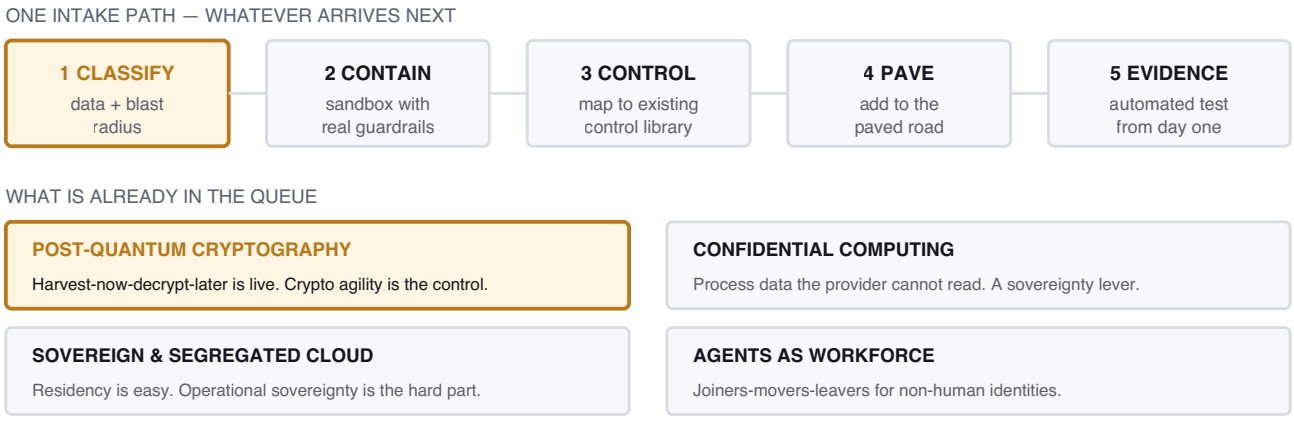
CONCENTRATION RISK, SHARPENED

Frontier models are not portable. “Switch providers” is not a like-for-like swap: prompts, tuning, guardrails and evaluation results are all provider-specific to some degree. When you assess concentration and exit under Chapter 17 and RB-11, treat AI dependencies as harder to unwind than compute or storage, and say so explicitly in the assessment rather than letting it be assumed away.

17 · Emerging Technology: Build the Intake

AI is the current example. Before it, containers. Before that, cloud itself. A firm that runs a bespoke governance programme for each new technology is permanently eighteen months behind.

The durable capability is a repeatable intake path, the same five steps regardless of what arrives. RB-10 is the procedure; this chapter is why it matters and what is already in the queue.



> The intake is the capability. The list beneath it changes every eighteen months; the five steps do not.

Post-quantum cryptography

This is the one with no discretion and a long lead time. Encrypted traffic captured today can be decrypted once a cryptographically relevant quantum computer exists, which means any data with a long confidentiality life — mortgage records, customer identity data, long-dated contracts, key material, is *already* exposed to harvest-now-decrypt-later. NIST has published the standards.

The migration isn't primarily a cryptography problem. It is an inventory problem. Almost no large institution can currently answer "where do we use RSA and ECC, in what, terminated by whom, and who owns each?" The operating model requirement is **crypto agility**: knowing every place a primitive is used and being able to change it without a rewrite. RB-12 is the inventory procedure, and it is the multi-year part, start it before you need it.

Confidential computing

Hardware-isolated enclaves let workloads process data the provider can't read. Genuinely useful for the most sensitive workloads and for sovereignty conversations. Worth understanding and having a position on before a supervisor asks whether you have considered it.

Sovereign and segregated cloud

Requests are increasing and firms consistently underestimate them. Residency is a region constraint. Operational sovereignty — which humans, in which jurisdictions, under which legal compulsion, can access systems, is much harder and materially restricts which managed services you can consume. Decide which you're actually being asked for.

PART FOUR

Runbooks

RUNBOOK RB-01

Account and Landing Zone Vending

PURPOSE	Deliver a team a production-ready, compliant cloud environment without a human security review.
TRIGGER	A team requests a new account, subscription or project.
OWNER	Platform engineering. Second line is *not* in this path.
TARGET SLA	Automated: under 1 hour. Manual fallback: 2 business days.
EVIDENCE	Vending request record; post-provision control test result; guardrail attachment confirmation.

1. **Capture the request against a service catalogue entry.**
Required fields: requesting team, named technical owner, named business owner, data classification (highest the environment will hold), environment type (prod / non-prod), cost centre, expected regions, and whether the workload is customer-facing.

2. **Validate automatically, reject fast.**
Unknown cost centre, missing owner, or a data classification the requesting team is not authorised to hold, reject with the specific reason. Do not queue these for human triage; the requester can fix them faster than you can.

3. **Provision from code.**
Account or subscription created via the vending pipeline. No console-created environments, ever — a manually created account is outside the guardrail hierarchy and nobody will notice for months.

4. **Attach the organisational guardrails.**
Placement in the correct organisational unit or management group, which carries the preventive policy set from Appendix A. This must happen before any workload identity exists in the account.

5. **Apply the baseline.**
Logging to the central immutable store, network topology and egress control, identity federation, secrets manager, encryption defaults, cost budget with alerting, and tagging.

6. **Run the post-provision control test immediately.**
Do not assume provisioning succeeded because the pipeline exited zero. Assert the guardrails are attached and effective by testing an action that should be denied. Record the result as the environment's birth certificate.

7. **Register the environment.**
Write it to the estate inventory with both owners, classification and creation date. This inventory is what every later control test enumerates — an environment missing from it is invisible to your entire assurance model.

8 **Hand over.**
Provide the team with pipeline access, the paved-road documentation, and the name of the exception process should they need to deviate.

FAILURE MODE TO DESIGN AGAINST

The most common defect is a vended environment where guardrails were attached *after* a workload identity already existed, leaving a window in which a non-compliant resource was created and then grandfathered. Step 4 must strictly precede step 5, and step 6 must actually test rather than inspect configuration.

Review this runbook whenever the guardrail baseline changes, and at least annually.

RUNBOOK RB-02

Guardrail Change Management

PURPOSE	Add, modify or remove a preventive guardrail without breaking production.
TRIGGER	New standard, incident lesson, provider feature change, or an exception pattern that has become common enough to warrant a policy change.
OWNER	Platform engineering implements. Second line approves the intent.
TARGET SLA	Standard: 4 weeks end to end. Emergency: same day with retrospective approval.
EVIDENCE	Change record, impact analysis output, approval, deployment log, post-deployment test.

1. **State the control intent in one testable sentence.**
   If it cannot be expressed as a query over resource state, it's not ready to be a guardrail.
2. **Run impact analysis against the current estate.**
   Query how many existing resources would violate the proposed guardrail. This number decides everything that follows. Zero means you can deploy in enforce mode. Non-zero means you have a remediation programme first.
3. **Deploy in audit mode.**
   Non-blocking, logging violations only. Leave it for a full business cycle — at minimum two weeks, and long enough to capture month-end and any batch processing.
4. **Review audit findings and revise.**
   Expect false positives on the first pass. A guardrail that fires on legitimate architecture will be exempted into uselessness, so fix the rule rather than the exemption list.
5. **Remediate or grant time-boxed exceptions**
   for the remaining legitimate violations, using RB-03. Every one gets an expiry date before enforcement begins.
6. **Enforce progressively.**
   Non-production first, then a low-criticality production group, then the estate. Never organisation-wide in one step, whatever the pressure.
7. **Verify enforcement empirically.**
   Attempt the prohibited action in a test environment and confirm it is denied. Configuration showing "enforce" is not proof of enforcement.
8. **Update the control library and add the automated control test.**
   A guardrail with no test does not produce evidence and will be invisible at examination.

EMERGENCY PATH

For an active exploited vulnerability, deploy in enforce mode immediately with the CISO or delegate approving verbally, and complete steps 1, 2 and 8 retrospectively within five business days. Record it as an emergency change — a pattern of emergency changes is itself a finding worth surfacing.

RUNBOOK RB-03

Exception Request, Approval and Expiry

PURPOSE	Permit a justified deviation from standard without allowing it to become permanent.
TRIGGER	A team cannot meet a control and has a business reason.
OWNER	Second line owns the register and the decision. Business owner accepts the risk.
TARGET SLA	Decision within 10 business days. Deliberately slower than the paved road.
EVIDENCE	Register entry, risk assessment, approval with named approver, expiry date, closure record.

1. **Require a complete request.**
Control being deviated from, precise technical reason, business impact of compliance, proposed compensating controls, requested duration, and the named business owner who will accept the risk. Incomplete requests are returned, not triaged.

2. **Check whether the paved road is the actual problem.**
If three or more teams have requested the same exception, this is a road defect. Route it to RB-02 as a guardrail or paved-road change instead of granting a fourth exception.

3. **Assess the risk.**
Second line assesses residual risk with compensating controls applied, and maps it against stated risk appetite.

4. **Route approval by residual risk, not by convenience.**
Within appetite: second-line head. Outside appetite: named executive, with the risk committee informed. Materially outside: risk committee decision. Nobody approves an exception for their own workload.

5. **Set an expiry date — always.**
Maximum 12 months, and shorter where a fix is in flight. “Permanent” is not an available option. If the deviation is permanent, it’s a standard change, so change the standard.

6. **Suppress the related finding, scoped precisely.**
Suppress for the specific resource and control, never estate-wide, and make the suppression itself expire on the same date. An exception that outlives its suppression creates noise; a suppression that outlives its exception hides a real risk.

7. **Notify before expiry.**
Automated notice at 60, 30 and 7 days to both owners. On expiry with no renewal, the suppression lifts automatically and the finding reappears.

8. **Report the register monthly.**
Total open, opened and closed this period, and — the number that matters — the count past expiry, by owner.

THE SINGLE MOST IMPORTANT LINE IN THIS BOOK

Renewal must require the same approval as the original request. If renewal is a lighter-touch process, every exception becomes permanent within two cycles, and your control framework becomes a description of what you wish were true.

RUNBOOK RB-04

Cloud Security Incident Response

PURPOSE	Contain, investigate and recover from a cloud security incident while preserving evidence and meeting notification obligations.
TRIGGER	Detection of unauthorised access, data exposure, credential compromise, cryptomining, or destructive action.
OWNER	Security operations leads. Product team supports. Legal and regulatory reporting engaged early.
TARGET SLA	Triage 15 min. Containment decision 1 hour. Regulatory assessment within your jurisdiction's clock — know it before you need it.
EVIDENCE	Timeline, actions log, preserved artefacts, decision record, post-incident review.

1. **Declare and start a timeline immediately.**
One channel, one incident commander, one running log with timestamps. Reconstructing this afterwards is impossible and it is what regulators ask for.

2. **Preserve before you contain.**
Snapshot affected volumes, export relevant logs to the immutable store, capture memory if the workload supports it. Containment frequently destroys the evidence that explains scope, and scope is what determines notification.

3. **Contain by identity first, network second.**
In cloud the most effective containment is usually revoking credentials, attaching an explicit deny policy, and invalidating active sessions, not network isolation. Remember that revoking a role does not terminate sessions already issued; you must explicitly invalidate them.

4. **Determine blast radius from the control plane log.**
Enumerate every action taken by the compromised identity across the full retention window, not just the alerting period. Assume lateral movement until disproven.

5. **Assess data impact explicitly and early.**
What data was accessible, what was actually accessed, and what was exfiltrated are three different questions with three different answers. Regulatory notification usually turns on the second and third.

6. **Engage legal and regulatory reporting on a clock.**
Notification deadlines are short and start earlier than teams expect. Bring them in during containment, not after recovery.

7. **Recover through the pipeline, not by hand.**
Rebuild from known-good infrastructure code. Manual repair of a compromised environment leaves persistence you will not find.

8 **Rotate everything the identity could reach.**
Not only the compromised credential — anything it had access to must be treated as exposed.

9 **Run a blameless post-incident review within 10 business days.**
Output is a set of owned actions with dates, and at least one candidate guardrail for RB-02. An incident that produces no control change has been wasted.

RUNBOOK RB-05

Break-Glass Access

PURPOSE	Provide emergency administrative access when normal identity paths are unavailable, without creating a standing back door.
TRIGGER	Identity provider outage, federation failure, or an incident requiring access beyond normal elevation.
OWNER	Platform engineering holds custody. Second line reviews every use.
TARGET SLA	Available within 15 minutes. Review within 1 business day of use.
EVIDENCE	Credential retrieval record, session log, justification, review sign-off.

1. **Maintain the accounts correctly.**
A small fixed number, one per cloud organisation. No federation dependency. That's the entire point. Hardware MFA. No email address that depends on the systems they exist to recover.

2. **Store under dual control.**
Credential and second factor held separately, by different people, in a way that requires two individuals to combine. A break-glass credential one person can use alone is just an unmonitored administrator account.

3. **Alarm on retrieval, not on use.**
Alerting must fire when the credential is retrieved from custody, to the CISO and the platform lead, out of band. By the time it is used you have already lost the early warning.

4. **Record justification at retrieval.**
Incident reference, requester, approver, expected duration.

5. **Timebox the session and monitor it live**
where the incident allows. Every action under a break-glass identity is high-consequence by definition.

6. **Rotate immediately after use.**
The credential is burned. Reissue, re-split custody, confirm the old one no longer authenticates.

7. **Review after every single use.**
Not quarterly. Second line reviews the session log against the stated justification and confirms every action was necessary and within scope.

8. **Test quarterly.**
An untested break-glass path fails on the day you need it — expired credentials, a rotated MFA seed, a custodian who left. The test is a deliberate retrieval and authentication, logged as a test.

RUNBOOK RB-06

Control Test Failure Triage

PURPOSE	Respond correctly when an automated control test fails — including when the test itself is wrong.
TRIGGER	A scheduled control test returns a failure, or returns nothing at all.
OWNER	Control testing triages. Resource owner remediates. Platform fixes road defects.
TARGET SLA	Triage 1 business day. Remediation per severity.
EVIDENCE	Test output, triage decision, remediation record, retest result.

1. **First question: did the test actually run?**
   A test that returns zero results is far more likely to be broken than to indicate a perfect estate. Treat “no failures and no population” as a failure of the test, and alert on it separately.

2. **Second question: is the test correct?**
   Before raising findings against teams, confirm the test reflects the current standard. Tests drift when standards change and nobody updates them; raising false findings destroys credibility quickly.

3. **Classify the failure.**
   Genuine control failure; known exception not correctly suppressed; test defect; or scope gap where a resource type is not covered.

4. **For a genuine failure, assign to the resource owner from the estate inventory.**
   If there is no owner, that’s a separate and more serious finding — an unowned production resource.

5. **Check for a road defect.**
   If several teams fail the same control, the paved road is producing non-compliant infrastructure. Fix the road first; individual remediation is treating a symptom.

6. **Remediate to severity.**
   Critical same day, high within 5 business days, medium 30, low next planned cycle. Escalate on breach to the owner’s management chain.

7. **Retest and record.**
   Closure requires a passing test result, not an assertion that it was fixed.

8. **Feed persistent failures upward.**
   A control that fails repeatedly across the estate is a candidate for a preventive guardrail via RB-02.

RUNBOOK RB-07

Examination and Audit Evidence Request

PURPOSE	Respond to a supervisory or audit evidence request accurately and without consuming an engineering quarter.
TRIGGER	Examination request, internal audit fieldwork, external audit, or client due diligence.
OWNER	Second line coordinates. Control testing produces. Nobody edits evidence.
TARGET SLA	Automated evidence within 2 business days. Bespoke analysis 10.
EVIDENCE	The request, what was provided, and when. Keep this; it becomes precedent.

1. **Translate the request into control IDs.**
   Requests arrive in the requester's vocabulary. Map to your control library first; this is where the mapping in Appendix E earns its keep.
2. **Establish the period and population precisely.**
   Which entities, which environments, which dates. Ambiguity here causes rework and, worse, the appearance of evasion.
3. **Pull from the evidence store, not from live systems.**
   Live state answers "is it compliant now". The question is almost always "was it compliant throughout", which only the historical store can answer.
4. **Include failures.**
   Provide the complete record including periods where the control failed, alongside the remediation record. A clean run with no failures invites scrutiny of the test, and selective disclosure is a far worse finding than an honest failure.
5. **Have the control owner review for accuracy, not for presentation.**
   Correct factual errors. Do not curate.
6. **Provide with context.**
   A short covering note: what the control is, how it is tested, what the population was, and any known limitations. Unexplained raw output generates more questions than it answers.
7. **Log the request and response.**
   The same evidence will be requested again by someone else. Build the library.
8. **Feed gaps into the backlog.**
   Anything you could not produce automatically is evidence debt. Add it to the control library as a gap with an owner.

NEVER

Never modify, backfill or reconstruct evidence to close a gap. If evidence does not exist for a period, say so plainly and explain what changed. Fabricated or retrospectively assembled evidence converts a control finding into a conduct issue, and the difference in consequence is enormous.

RUNBOOK RB-08

AI System Intake and Approval

PURPOSE	Approve an AI system for production with proportionate control, and keep the model inventory complete.
TRIGGER	Any proposal to use a model against firm or customer data — including a vendor enabling an AI feature in an existing product.
OWNER	Model risk / second line approves. Product team owns the system. Security reviews the boundary.
TARGET SLA	Low risk 10 business days. High risk per model risk governance.
EVIDENCE	Inventory entry, risk tier, retrieval authorisation test, evaluation results, approval.

1. **Register it before assessing it.**
Inventory entry first, using Appendix C. An unregistered system in production is the failure this process exists to prevent, and registration must not depend on approval succeeding.

2. **Identify the deployment pattern**
— A (embedded in SaaS), B (managed endpoint) or C (self-hosted weights). This determines who owns which control.

3. **Classify the data.**
What goes into prompts, what the system can retrieve, and what appears in outputs. Prompt content is frequently more sensitive than the team assumes.

4. **Assign a risk tier.**
Consider decision impact on customers, regulatory exposure (creditworthiness and similar uses are high-risk under the EU AI Act), data classification, whether output reaches customers unreviewed, and whether the system has tools. Any system with tools goes to RB-09 as well.

5. **Confirm the data path.**
Private networking where the pattern allows, customer-managed keys, contractual position that prompts are not used for provider training, and a documented data residency position.

6. **Require prompt and completion logging**
with the log store classified at the sensitivity of the most sensitive possible prompt content, and a defined retention aligned to records obligations.

7. **Test the retrieval boundary with a low-privilege account.**
Verify that a user cannot obtain content they are not entitled to. This is a live test, not a design review. It is the most commonly failed step and the most commonly skipped one.

8 **Require an evaluation harness**
covering accuracy on representative cases, refusal behaviour, prompt injection resistance, and bias where the use case warrants it. Results are recorded and re-run on provider model changes.

9 **Define the human oversight position.**
Who reviews output, in which cases, and what the fallback is when the model is unavailable or wrong. High-risk uses require this to be specific.

10 **Set cost and rate limits with alerting**
before go-live.

11 **Approve with a review date.**
Foundation models change underneath you; approval is not indefinite.

RUNBOOK RB-09

Autonomous Agent Deployment Approval

PURPOSE	Approve an AI system that can take actions, treating it as a privileged identity rather than as software.
TRIGGER	Any AI system with tool access, function calling, or the ability to make changes to a system.
OWNER	Security approves the permission model. Business owner accepts operational risk.
TARGET SLA	15 business days. Not expedited.
EVIDENCE	Tool classification, permission grant, gate configuration, injection test results, tool-call log sample.

1. **Complete RB-08 first.**
   This runbook is additional, not alternative.
2. **Enumerate every tool the agent can invoke.**
   Including indirectly, through other services it calls. An incomplete tool list invalidates everything downstream.
3. **Classify each tool by blast radius and reversibility.**
   Read-only; write-reversible; write-irreversible; money movement; external communication. Anything you cannot classify is treated as irreversible.
4. **Issue a dedicated identity.**
   Never a shared service account, never a human's credentials. Actions must be attributable to this agent.
5. **Scope permissions to the tool list exactly.**
   Derive the permission set from step 2 and nothing else. Resist convenience grants; they are how an agent that reads tickets ends up able to delete them.
6. **Implement a human approval gate**
   on every irreversible, money-movement and external-communication tool. Permanent, not a pilot-phase measure.
7. **Verify authorisation is deterministic.**
   Confirm by inspection that model output does not itself grant access, that a deterministic component decides whether a requested action is permitted. If the model's response is the authorisation, prompt injection is a full bypass and the design must change.
8. **Test prompt injection against the real tool set.**
   Attempt to induce an unauthorised tool call through untrusted input the agent will encounter — a document, an email, a web page, a ticket comment. Record what was attempted and what happened.
9. **Enable tool-call logging with reasoning**
   and confirm the log is queryable for incident reconstruction.

10 **Set an operating envelope.**
Rate limits, spend limits, and a defined kill switch with a named person able to operate it and a tested procedure.

11 **Register the agent as a non-human identity**
with an owner, a review cycle, and a retirement process. Agents are joiners; make sure they can also be leavers.

RUNBOOK RB-10

New Technology Intake

PURPOSE	Bring an unfamiliar technology into governed use quickly, using one repeatable path rather than a bespoke programme.
TRIGGER	A team wants to use a service, pattern or product with no existing control position.
OWNER	Second line classifies. Platform paves. Control testing instruments.
TARGET SLA	Sandbox within 10 business days. Production position within 90 days.
EVIDENCE	Classification record, sandbox constraints, control mapping, paved-road addition, first test result.

1. **Classify by data and blast radius.**
   What data will it touch, and what is the worst thing it could do. These two answers determine the entire response, and both are answerable in an afternoon.
2. **Contain it in a real sandbox.**
   An isolated environment with genuine guardrails, non-production data only, egress controlled, time-limited. A sandbox without guardrails is just an ungoverned environment with a friendly name.
3. **Map to the existing control library before writing anything new.**
   Most “new” technology needs existing controls applied differently rather than novel controls. Identify the genuine gaps — there are usually two or three, not thirty.
4. **Write control positions only for the genuine gaps,**
   and add them to the library with owners.
5. **Add it to the paved road**
   with secure defaults, so the second team to adopt it does not repeat the first team’s work.
6. **Instrument it from day one.**
   Automated control tests before general availability, not after. Retrofitting evidence onto an adopted technology is how evidence debt accumulates.
7. **Publish the position.**
   Approved with constraints, approved for a use class, or not approved with a reason. Silence is read as permission.

RUNBOOK RB-11

Cloud Exit and Stressed Exit Test

PURPOSE	Demonstrate that a material workload can be recovered or relocated if the primary provider becomes unavailable or unusable.
TRIGGER	Annual, on material architecture change, or on supervisory request.
OWNER	Business service owner. Platform supports. Second line observes and challenges.
TARGET SLA	Annual per material service. Plan refresh on material change.
EVIDENCE	Exit plan, test scope, execution log, achieved RTO/RPO, gap register.

1. **Scope to a material business service, not to infrastructure.**
   Exit is meaningful at the level of a service the firm provides customers, not at the level of a database.

2. **Distinguish the scenarios.**
   Provider failure, regional failure, contractual or regulatory forced exit, and provider-initiated service withdrawal are different problems with different timelines. Test the one your plan claims to handle.

3. **Enumerate the dependencies honestly, including managed services.**
   This is where plans break. A workload using a proprietary managed database, queue and identity service is not portable regardless of what the plan asserts, and AI dependencies are the least portable of all.

4. **State the target RTO and RPO and where they came from.**
   They should derive from the business impact assessment, not from what the architecture happens to achieve.

5. **Execute a real test.**
   A tabletop exercise is not an exit test. Restore into an alternative environment, run functional verification, and measure how long it actually took.

6. **Record the gap between plan and reality without softening it.**
   The value of the test is the gap. A test that reports full success usually means the scope was chosen to guarantee it.

7. **Feed gaps into architecture decisions.**
   If exit is not achievable within appetite, that's a risk acceptance requiring an executive decision, or an architecture change. It is not a documentation problem, and it should not be resolved quietly.

8. **Update the register of information**
   with current criticality, substitutability and exit position.

RUNBOOK RB-12

Cryptographic Inventory for PQC Migration

PURPOSE	Build the cryptographic inventory that post-quantum migration depends on, and establish crypto agility.
TRIGGER	Start now. Harvest-now-decrypt-later exposure is already accruing against long-lived data.
OWNER	Security architecture owns the inventory. Application owners supply their own entries.
TARGET SLA	Initial inventory 6 months. Continuous thereafter.
EVIDENCE	Inventory with coverage percentage, prioritised migration plan, agility assessment.

1. **Accept that this is an inventory problem, not a cryptography problem.**
The algorithms are standardised. Almost no institution can currently say where its cryptography lives, and that is the multi-year part.

2. **Enumerate mechanically first.**
TLS termination points, certificates and their issuers, code-signing keys, KMS and HSM key material, VPN and tunnel endpoints, database and storage encryption, JWT and token signing, and application-embedded cryptography. Automated discovery gets you most of the way on the first four.

3. **Record what actually matters per entry.**
Algorithm and key length, purpose, owner, where implemented, whether it is provider-managed or yours, and — critically — the confidentiality lifetime of the data it protects.

4. **Prioritise by confidentiality lifetime, not by system criticality.**
Data that must stay confidential for fifteen years is at risk today; a session token that expires in an hour is not. This inverts the usual prioritisation and is the single most useful judgement in the exercise.

5. **Assess agility per entry.**
Could you change the algorithm without a rewrite? Hard-coded primitives and pinned certificates are the blockers, and finding them is most of the value even before any migration begins.

6. **Fix agility before migrating.**
An estate that can change algorithms can migrate whenever it needs to, including again when guidance changes. Chasing the migration without agility means doing it twice.

7. **Track provider readiness**
for the managed services you consume, and record which parts of the migration are simply not yours to perform.

8. **Report coverage, not completion.**
“We have inventoried 60% of TLS endpoints” is honest and actionable. “PQC migration is in progress” is neither.

RUNBOOK RB-13

Continuous Control Assurance Review

PURPOSE	Confirm that control testing is genuinely running, covering the estate, and would be noticed if it stopped — so you can evidence controls between audits, not only at them.
TRIGGER	Monthly. Also after any change to the test platform, cloud organisation structure, or logging pipeline.
OWNER	Control testing runs it. Second line reviews the output.
TARGET SLA	Monthly, within 5 business days of period end.
EVIDENCE	Coverage report, staleness report, drift report, and the review sign-off.

1. **Check test liveness first, before looking at any result.**
   For every control test, when did it last report? Anything outside its expected window is treated as a **control failure**, not a tooling problem. A test that stopped four weeks ago has been asserting nothing for four weeks while showing green.

2. **Verify each test has a dead-man's switch.**
   Not just that it ran, but that its absence would page somebody. A test whose silence is silent is worth substantially less than the effort spent building it.

3. **Measure coverage against the estate inventory, not against itself.**
   Take the authoritative list of accounts and resources from RB-01 and calculate what percentage each test actually reached. Falling coverage with a stable pass rate is the classic signature of a test that has quietly lost access.

4. **Reconcile the population.**
   Compare accounts the tests can see against accounts that exist in the cloud organisation. Any account visible to the provider but invisible to testing is an urgent finding — it is unmonitored production.

5. **Check for boundary drift.**
   New accounts, detached or modified organisation policies, changed organisational unit membership, altered logging destinations. Each of these invalidates other tests for the resources affected.

6. **Look for suspiciously perfect results.**
   Any test with a 100% pass rate over a long period and no failures ever deserves scrutiny. Confirm it can fail by testing it against a deliberately non-compliant resource in a sandbox.

7. **Confirm evidence is landing and is immutable.**
   Spot-check that results written 30, 90 and 365 days ago are still present, unaltered, and retrievable. Retention that was never verified is not retention.

8 **Produce the assurance statement.**
One page: tests live, coverage percentage, population reconciliation, drift observed, and any period during which a control could not be evidenced. Say so plainly where evidence is missing, a documented gap is defensible and a discovered one is not.

THE POINT OF THIS RUNBOOK

Everything here exists because a control test that stopped running is visually identical to one that is passing. If you adopt only one thing from this manual beyond the exception expiry rule, make it step 1.

PART FIVE

Templates and Reference

A · The Preventive Guardrail Baseline

A starting set of organisation-level preventive controls. Deliberately short, these are the ones worth defending absolutely. Deploy via RB-02, never all at once.

#	Guardrail	Rationale and notes
1	No public object storage	The most common cause of publicly disclosed cloud data exposure. Enforce at organisation level with account-level public access blocks.
2	No unencrypted data stores	Storage, databases, disks, snapshots, backups. Snapshots and backups are the ones usually missed.
3	No internet-facing databases	Database services must not be assigned public endpoints under any circumstances.
4	Audit logging cannot be disabled or deleted	Protects your evidence base. Log destination must be write-only from the source account.
5	No long-lived static access keys	Federated, short-lived credentials only. The highest-value single control in cloud identity.
6	Region restriction	Resources only in approved regions. Underpins residency claims and reduces the estate you must monitor.
7	Guardrails cannot be detached	Prevent member accounts leaving the organisation or removing policy. Protects the whole hierarchy.
8	No disabling of security services	Threat detection, config recording and posture services cannot be turned off locally.
9	Root / global admin use blocked and alarmed	Break-glass only, per RB-05. Any use is an alert, not a log entry.
10	No unrestricted inbound (0.0.0.0/0) to management ports	SSH, RDP, database ports. Access via bastion or private connectivity only.
11	Immutable retention on the evidence store	Object lock or equivalent. Nobody — including administrators, can alter historical evidence.
12	Encryption key deletion protected	Mandatory waiting period and alarm. Key deletion is indistinguishable from destructive attack.
13	No sharing resources outside the organisation	Blocks snapshots, images and role trust being shared to unknown accounts.
14	Mandatory tagging for owner and classification	Untagged resources are unattributable, and unattributable resources cannot be remediated.
15	AI model endpoints only via approved path	Blocks direct calls to unapproved model providers, forcing use of the governed endpoint. Increasingly the guardrail that prevents pattern-A sprawl.

SIZING

If your preventive set grows past roughly twenty, you are using guardrails where defaults belong, and you will generate exception volume you cannot service. Push the marginal ones down into the paved road.

B · Exception Register Schema

Implement in whatever system you already use. The fields matter more than the tool.

Field	Notes
Exception ID	Immutable, sequential.
Control ID	Links to the control library. Mandatory.
Scope	Specific resources, accounts or workloads. Never “all environments”.
Reason	The technical constraint, stated concretely.
Business impact of compliance	What would happen if it were not granted. Forces honesty.
Compensating controls	What reduces residual risk, and who verifies each.
Residual risk rating	Post-compensation. Drives approval level.
Technical owner	Named individual.
Risk-accepting owner	Named individual with authority to accept. Not the same person.
Approver and date	Per the routing in RB-03.
Granted date / Expiry date	Expiry mandatory, maximum 12 months.
Suppression reference	The specific finding suppression, expiring on the same date.
Remediation plan	What removes the need, and by when.
Status	Open / Renewed / Expired / Closed-remediated / Closed-accepted.
Renewal count	Report this. Three or more renewals means the standard or the road is wrong.

C · AI Model and System Inventory

Field	Notes
System ID / name	The business system, not the model.
Deployment pattern	A (SaaS-embedded) / B (managed endpoint) / C (self-hosted).
Model and version	Including provider. Changes when the provider updates, track it.
Business owner / technical owner	Named individuals.
Purpose	One sentence on what it does.
Data in prompts	Classification of what is sent. Usually underestimated.
Retrieval sources	What it can read, and how per-user authorisation is enforced at retrieval.
Output destination	Internal only / customer-facing / feeds an automated decision.
Has tools?	Yes triggers RB-09. List tools and their blast-radius class.
Risk tier	Your tiering, plus EU AI Act classification where relevant.
Model risk category	Where it sits under your SR 11-7 equivalent framework.
Human oversight	Who reviews what, and the fallback when unavailable.
Evaluation harness	Location, coverage, last run, last result.
Prompt log location / retention	Including the classification applied to the log store.
Approval and review date	Approval is never indefinite.

D · RACI for the Operating Model

Activity	Product	Platform	2nd line	Audit	Exec
Define control standard	C	C	**A/R**	I	I
Implement guardrail	I	**A/R**	C	I	—
Vend environment	C	**A/R**	I	—	—
Operate workload securely	**A/R**	C	I	—	—
Remediate finding	**A/R**	C	I	I	—
Approve exception (in appetite)	C	I	**A/R**	I	I
Approve exception (out of appetite)	C	I	R	I	**A**
Run control tests	I	C	**A/R**	I	—
Assert control effectiveness	—	—	C	**A/R**	I
Approve AI system	R	C	**A**	I	I
Approve agent permissions	R	C	**A**	I	I
Incident command	C	C	**A/R**	I	I
Accept residual risk	C	—	R	I	**A**

A = accountable (one only) · R = responsible · C = consulted · I = informed

E · Control-to-Framework Mapping

Worked rows showing the structure from Chapter 9. Map once; inherit outward.

ID	Statement	Test	Maps to
CLD-001	All object storage denies public access at account and bucket level.	Daily query, full population.	CRI PR.AC; 800-53 AC-3, AC-6; CIS 2.1; PCI 1.3
CLD-002	All data stores encrypt at rest with a customer-managed key where classification is Confidential or above.	Daily query, full population.	CRI PR.DS; 800-53 SC-28; PCI 3.5
CLD-007	No IAM principal holds a static access key older than the rotation standard.	Daily query, full population.	CRI PR.AC; 800-53 IA-5
CLD-011	Control-plane audit logging is enabled in every account and delivered to the immutable store.	Daily; alert on gap in delivery.	CRI DE.AE; 800-53 AU-2, AU-9
CLD-020	Break-glass credential use generates an alert and a review record within one business day.	Event-driven; monthly reconciliation.	CRI PR.AC; 800-53 AC-2(7)
AI-001	Every production AI system appears in the model inventory with an owner and risk tier.	Monthly reconciliation vs endpoint telemetry.	Internal AI policy; EU AI Act Art. 9 where high-risk
AI-004	Retrieval enforces per-user authorisation at query time.	Quarterly live test with low-privilege account.	CRI PR.AC; 800-53 AC-3
AI-006	No agent may invoke an irreversible tool without recorded human approval.	Continuous; sample tool-call log monthly.	Internal AI policy; 800-53 AC-6(9)

F · Board Reporting Pack

One page. Six numbers with trend, three commentary prompts, and the exceptions table. Anything longer will not be read.

Metric	This period	Trend	Commentary prompt
Paved road adoption (%)			If falling, what is driving teams off the road?
Median time to compliant environment			Is this competitive with the unofficial alternative?
Exceptions past expiry (count)			Named owners for each. Why has renewal not occurred?
MTTR — critical / high			What proportion was auto-remediated?
Controls with no automated test			Which are highest risk, and when are they instrumented?
Unregistered AI systems / NHIs found			How were they found, and what closes the gap?

Three standing questions for the committee: What changed in our risk position this period, and why? What did we learn from an incident or a near miss? What are we deliberately not doing, and what is the accepted consequence?

G · Implementation Plan

First 90 days

Do not attempt breadth. Establish the spine: publish the control standard (Chapter 2); deploy the first tranche of preventive guardrails in audit mode (RB-02); build environment vending (RB-01) even if crude; stand up the exception register with expiry enforcement (RB-03); and instrument three controls end to end so the evidence pipeline exists in miniature. Also register every AI system already in production — you will find more than expected, and that finding is useful.

Year one

Move guardrails from audit to enforce, progressively. Drive paved road adoption above 70% for new workloads. Get the exception register clean, with zero past expiry. Build the control library with framework mapping. Stand up AI intake (RB-08) before AI adoption outruns it. Separate second-line challenge from tooling operation if they are currently the same team. This is organisational and takes the full year.

Year two

Automate evidence for the majority of controls. Give internal audit self-service access to the evidence store. Run the first stressed exit test (RB-11) and report the gap honestly. Begin the cryptographic inventory (RB-12). Establish agent governance (RB-09) before agentic systems reach production, not after.

Year three

Exit tested and gaps either closed or explicitly accepted at executive level. Non-human identity governance operating with reviews and a retirement process. PQC migration underway on a prioritised basis with crypto agility established. New technology entering through RB-10 as routine rather than as an event. At this point the model is self-sustaining, and the work becomes maintenance and adaptation.

> **THE MOST COMMON SEQUENCING ERROR**
>
> Attempting evidence automation before the paved road exists. If workloads are still built inconsistently, every control test needs bespoke handling per environment and the automation never converges. Consistency first, then measurement.

H · Glossary

Blast radius	The scope of damage an action or compromise could cause. Used here to classify agent tools and to scope new technology.
Confused deputy	A system with high privilege performing an action on behalf of a lower-privileged requester without checking their entitlement. The core RAG risk.
Control test	A scheduled automated job asserting a control's state across the whole population and recording the result immutably.
Crypto agility	The ability to change cryptographic primitives without re-engineering. The actual prerequisite for PQC migration.
Guardrail	A preventive control at organisation level making a prohibited action impossible rather than detectable.
Harvest-now-decrypt-later	Capturing encrypted data today to decrypt once quantum capability exists. Why long-lived confidential data is already exposed.
Landing zone	A pre-configured, compliant cloud environment with guardrails, logging, identity and networking already in place.
NHI	Non-human identity — service principals, roles, managed identities, API keys. The majority of your identity population.
Operational sovereignty	Control over which humans, in which jurisdictions, under which legal compulsion, can access systems. Distinct from data residency.
Paved road	The supported, pre-approved way to build and deploy, made faster than the alternatives.
Prompt injection	Untrusted content causing a model to take unintended action. Privilege escalation once the model has tools.
RAG	Retrieval-augmented generation, supplying retrieved documents to a model at inference time.
SCP	Service control policy, an organisation-level permission boundary that cannot be overridden locally. Equivalents exist on all major providers.

I · Regulatory Reference Map

Named rather than paragraph-cited, so you can find the current version. Confirm applicability with your own counsel — this is a navigation aid, not an obligations register.

Reference	Relevance to this manual
FFIEC IT Handbook	US interagency expectations on architecture, operations and outsourcing. Baseline for examination scope.
OCC Heightened Standards	Risk governance framework for large national banks. Underpins the three-lines model in Chapter 3.
SR 11-7	US supervisory guidance on model risk management. The starting point for AI governance (Chapter 13).
DORA (EU)	ICT risk, third-party dependency, register of information, exit and resilience testing. Drives RB-11.
EBA outsourcing guidelines	Material outsourcing definitions, register, and audit rights.
FCA / PRA operational resilience	Important business services, impact tolerances, scenario testing. Shapes exit test scoping.
EU AI Act	Risk-tiered obligations. Creditworthiness assessment among high-risk uses. Phased timelines.
ECOA / Regulation B	Adverse action reasons. Constrains opacity in credit decisioning (Chapter 13).
GLBA Safeguards Rule	US customer information protection obligations.
PCI DSS	Cardholder data environment. Maps into the control library rather than sitting beside it.
NIST SP 800-53 / CSF	Common control vocabulary for the mapping in Appendix E.
CRI Profile	Financial-sector control framework designed to reduce duplicate questionnaire effort.
NIST PQC standards	Standardised post-quantum algorithms. Basis for RB-12 planning.

J · AI Decision Delegation Matrix

One row per security decision you have delegated, or are considering delegating, to an AI system. Reviewed quarterly. Chapter 15 explains the ladder and the four tests.

Field	Notes
Decision	Stated as a decision, not a capability. “Assign initial severity to a cloud posture finding”, not “AI for alerts”.
Current rung	1 Observe / 2 Recommend / 3 Act with approval / 4 Act and notify / 5 Autonomous.
Reversible?	Yes / No / Partially, with the reversal mechanism named.
Error detectable?	How a wrong decision surfaces, and after how long. “Unfalsifiable” is a valid and disqualifying answer.
Volume	Decisions per day. Drives whether human review is real.
Blast radius	Worst credible consequence of a systematically wrong decision, not a single one.
Human reviewer	Named role. Blank at rung 3+ is a finding.
Override rate	Percentage of AI outputs the human changed, last period. **Near zero at rung 3 means oversight has stopped functioning.**
Accountable owner	The person who owns the outcome. Never the model, never the vendor, never “the platform”.
Rung change history	When it moved, who approved, and on what evidence. Rungs must move by decision, not by drift.

DO NOT RECORD THESE AS DELEGABLE

Accepting risk · approving an exception · signing a regulatory attestation · concluding a control is effective · deciding an incident is notifiable · granting privileged access. Accountability for these is personal and non-transferable regardless of model capability.

K · Tool Consolidation Risk Assessment

Complete before consolidating onto a platform, and again at each renewal. Chapter 8 gives the argument on both sides.

Question	What a good answer looks like
What privilege does this platform hold?	Enumerated: agent privilege per host, API scopes, data read, execution capability. If the answer is “full”, say so plainly.
What is the blast radius if the vendor is compromised?	Stated as a scenario, not a rating. Which systems, which data, what the attacker could do with the platform’s own privilege.
What do we lose visibility of if it is offline?	The blind window. Include the vendor’s own outage history.
What independent signal survives?	Name it. “None” is a valid answer and an escalation.
Where does control evidence live?	Must not be solely inside the platform being evidenced.
Does break-glass depend on it?	If yes, redesign before proceeding.
What is the exit cost and time?	Realistic, including detection content, integrations and retraining. Compare against your exit expectations under RB-11.
Does this make it a critical third party?	If yes, it inherits concentration assessment, exit planning and resilience testing obligations. Confirm with second line.
What leverage do we retain at renewal?	Honest answer. Consolidation usually reduces it, which is a commercial risk with a security tail.
Decision and approver	Named. Consolidation past a materiality threshold is an executive risk decision, not an architecture preference.

L · Where This Gets Hard

The judgement calls I have not resolved for you, because I do not think they can be resolved in the abstract. Your sector, your appetite and your leverage should produce different answers than mine would.

How many preventive guardrails is too many? I have argued for ten to fifteen. I have also seen a firm run twenty-eight successfully and another drown in exception volume at twelve. What actually decides it is how good your paved road is: strong defaults let you hold fewer hard lines, because the marginal control gets absorbed rather than argued about. If your exception queue is growing faster than you can service it, you have your answer regardless of the count.

Where do you draw the line on second-line independence? I have been firm about this in Chapter 3 and I will be honest that it is easier to state than to fund. Small institutions frequently cannot staff a second line that does nothing operational, and the pragmatic answer — a first-line security function that operates tooling, with a genuinely separate risk function that challenges — is a compromise most examiners accept when the split is explicit. What they do not accept is the split existing only on an org chart.

How much evidence automation is enough? Every control with an automated test is the right target and almost nobody reaches it. The real question is which controls you are content to evidence manually forever, and that is a judgement about examination exposure and your own tolerance for a bad month. Write the list down; an explicit list of manually-evidenced controls is defensible, and a vague intention to automate everything is not.

At what rung do you actually stop delegating to AI? Chapter 15 gives you the ladder and the tests. It does not tell you where your organisation should sit, because that depends on your volume, your staffing and how much you trust your own override measurement. I would rather a firm sat confidently at rung 2 with honest metrics than claimed rung 4 with a rubber-stamp process.

When is consolidation worth the concentration? I have set out both columns in Chapter 8 without picking a side, and that is deliberate. A firm with a small team that cannot operate six products well is often safer consolidated, even accepting the concentration. A firm large enough to run diversity properly usually should. The mistake is consolidating for cost and telling yourself it was for security, then never doing the assessment in Appendix K.

How honest can you be with the board? Everything in this manual assumes you can report an uncomfortable number without it ending your tenure. That assumption does not hold everywhere, and where it does not hold, no framework will save you. If your organisation punishes the first honest metric, the metrics will stop being honest long before the risk stops being real. That is a cultural problem and it sits above anything in these pages.

USING THIS MANUAL

Everything here is meant to be taken apart. Rename the runbooks, renumber the controls, delete the sections that do not apply to your firm, and argue with the parts you disagree with, particularly where your risk appetite differs from the positions taken here. A manual adopted unchanged has usually not been read.