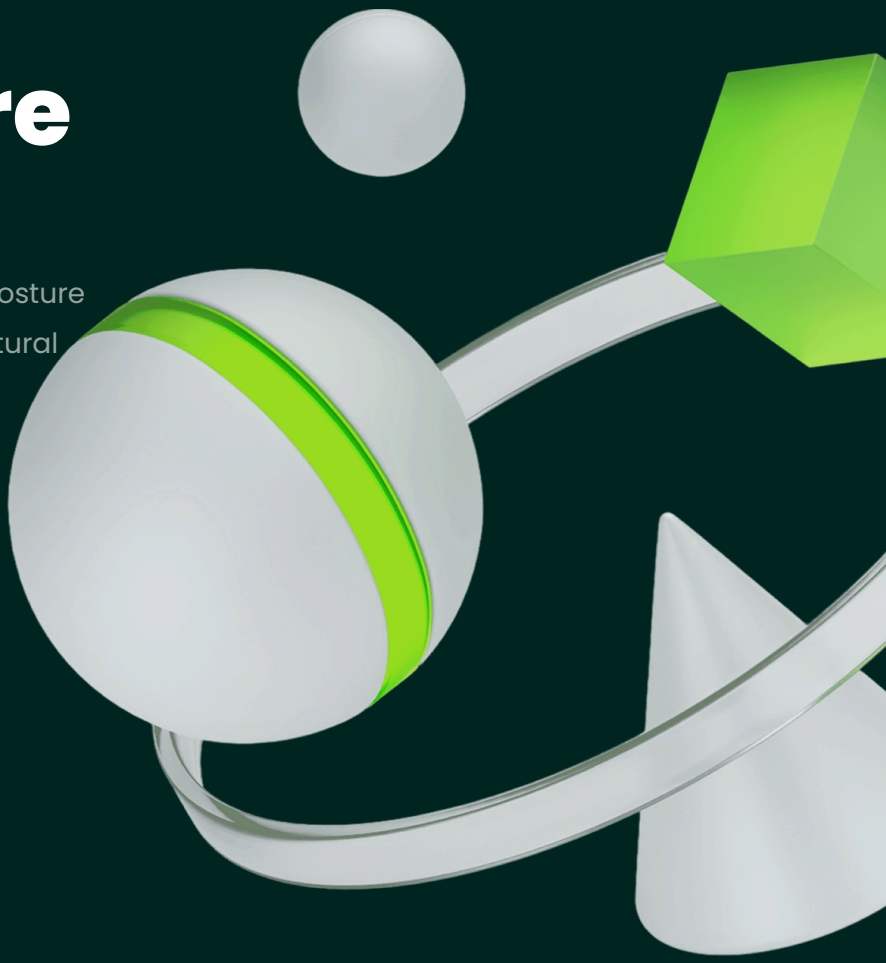
JupiterOne

EBOOK

CONTINUOUS CONTROLS MONITORING

In the Age of Ephemeral Infrastructure

How graph-native CCM transforms security posture validation — and makes audit readiness a natural by-product of strong security practice.



INTRODUCTION

The Controls Monitoring Gap

With cloud-native architectures, expanding AI implementations, and ever-shorter release cycles, today's technology ecosystems are more dynamic than ever before. Traditional strategies for validating security controls were designed for largely static environments — they simply can't keep pace.

The most common legacy model consisted of periodic manual checks — quarterly, annually, or right before an audit. This worked when infrastructure changed during infrequent maintenance windows. Modern enterprises incorporate thousands of asset types, changing so quickly that a manual inventory is stale before it's even complete.

Traditional controls validation offered only a moment-in-time snapshot. Putting that snapshot together reactively left security teams exhausted and scrambling. As soon as the audit was complete, control drift began again.

Continuous controls monitoring (CCM) enables proactive, real-time validation of an enterprise's security controls. Once CCM is implemented, compliance becomes a natural by-product of resilience — and audits become simple.

FROM REACTIVE TO

Continuous Assurance

WHAT THIS EBOOK COVERS

The architectural approach needed for successful CCM at enterprise scale — including controls-as-code, graph databases, three real-world control examples, and the three pillars of modern CCM.

DEFINITION

Continuous Controls Monitoring, Defined

CCM is the process of automatically and continuously evaluating the effectiveness of security controls across an organization's infrastructure — ensuring they're functioning as intended and delivering that assurance in real time. By identifying misconfigurations and control gaps, CCM makes it possible to correct them *before* an incident takes place.

CCM is profoundly different from legacy approaches to controls validation. Here's how they compare:

DIMENSION	MODERN CCM	LEGACY APPROACH
TEST FREQUENCY	Continuous: controls tested automatically and often	Periodic: typically right before audits
EVIDENCE	Automatic: collected and stored continuously	Manual: gathered by hand, stored in spreadsheets or GRC tools
TEST COVERAGE	Comprehensive: entire environment, always	Limited: only a sampled subset of systems
EASE OF TESTING	Simple: query-based, flexible, easy to customize	Labor-intensive: manual scripting and effort required
APPROACH	Proactive: drift is prevented, evidence always available	Reactive: test before audit, then drift resumes
RESULTS	Real-time insights in a drill-down dashboard	Static audit reports or spreadsheets

THE MECHANISM

How Modern CCM Works

In legacy testing, evidence was gathered by *sampling* — testers selected a set of transactions or changes and manually verified the control was applied. Coverage was never guaranteed, and there was no way to test the entire environment.

In modern CCM, **every control is tested**. For each control, a discrete, deterministic test returns a result validating whether the control is present or absent across the whole environment.

Negative Test
Searches for violations. The control is in place if **zero results** are returned.

Positive Test
Verifies the expected configuration is in place **everywhere it should be**.

Each test queries the organization's **cyber asset graph** — a complete model of every asset and the relationships between them. Tests run on automated schedules, with additional tests triggered when a known change has occurred.

THE FEEDBACK LOOP

The CCM dashboard shows current status for every control. Automated alerts notify the responsible owner of what needs remediation — creating a continuous feedback loop between infrastructure state and security posture.

ROUTINE TEST CADENCE

Tests can run daily or more frequently. Additional ad-hoc runs can be triggered any time a change is known to have taken place.

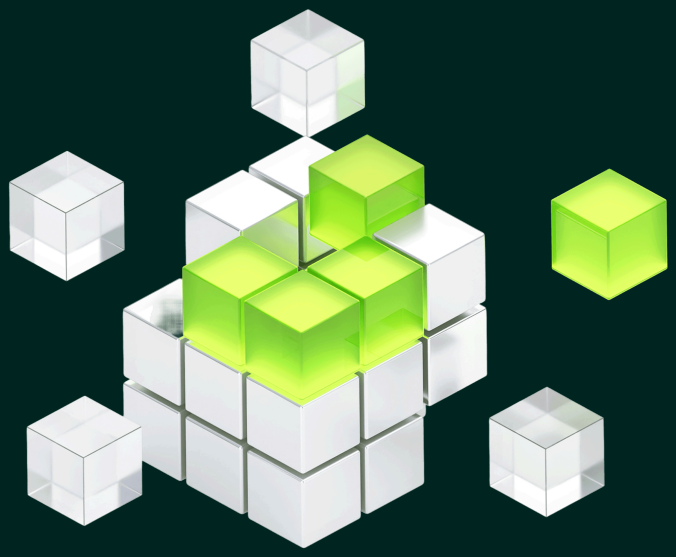
THE TECHNOLOGY FOUNDATION

A Graph Database Backbone

Conventional GRC tools are built on relational databases with rigid schemas that cannot model complex relationships between assets and controls at scale. Graph databases are designed to store entities *and* illustrate the relationships between them — a natural fit for cybersecurity.

A CCM solution built on a graph database can:

- Scale to massive amounts of enterprise data — every relationship between users, devices, apps, data, and cloud resources
- Answer granular questions based on the most detailed data model in existence, across the entire environment
- Test complex controls across multi-cloud environments using a highly flexible, extensible schema



GRAPH VS. RELATIONAL

A traditional database can't handle the extensive integrations, rapid schema growth, or complex traversals that characterize modern cloud environments. Graph databases are built for exactly this kind of fluidity.

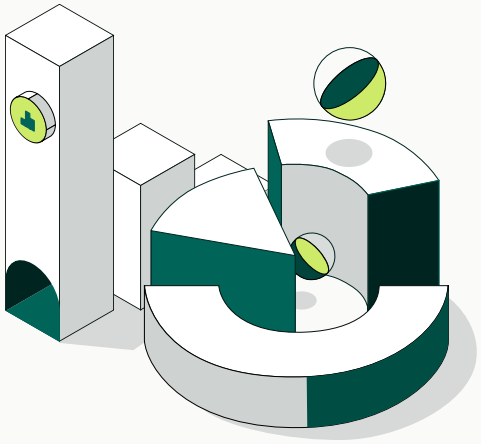
CONTROL EXAMPLE · 1 OF 3

Finding a Segregation of Duties Violation in GCP

In Google Cloud Platform, granting the same user both a **Service Account Administrator** and a **Service Account User** role creates a toxic combination. An attacker who gained access to that account could create backdoor service accounts, mint keys, and maintain persistent access even after credentials are rotated.

This SoD violation appears across SOX, ISO 27001, NIST CSF, and the CIS GCP Foundations Benchmark. In SQL, you'd need to cross-reference two separate policy tables for each instance — and repeat per failure. With CCM, a single query demonstrates the violation is absent across the *entire* environment:

```
-- Find users holding both SoD-violating roles
FIND google_user AS u
  THAT ASSIGNED google_iam_binding
    WITH role = "roles/iam.serviceAccountAdmin"
  THAT ASSIGNED google_iam_binding
    WITH role = "roles/iam.serviceAccountUser"
RETURN u.email, u.name
```



WHY THIS MATTERS

This violation is common — and commonly missed. CCM automates the test and proves the control holds across the entire environment. Zero results = zero violations.

RELEVANT FRAMEWORKS

SOX | ISO 27001 | NIST CSF | CIS GCP v4

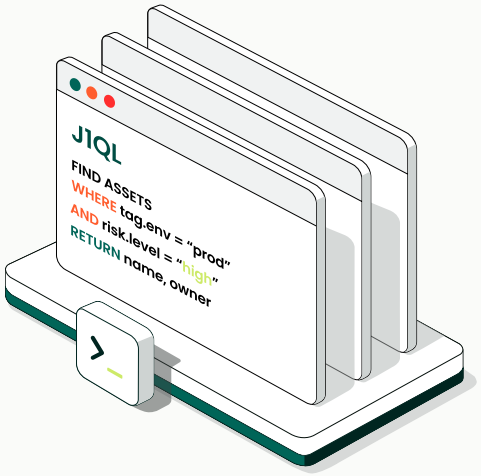
CONTROL EXAMPLE · 2 OF 3

Finding AWS Security Groups with Open Ports

Best practice limits inbound access to ports 22 (SSH) and 3389 (RDP) to specific IP ranges. But an engineer might open these ports while debugging, or grant a vendor temporary access — then forget to revert. Teams may also re-use overly permissive security group templates, widely propagating the misconfiguration across their AWS environment.

This exposes ports to any IP address on the open internet. Any attacker who gains access — through brute force or stolen credentials — gets remote administrative privileges, enabling lateral movement and access to cached credentials. Regex-based scanners miss port-interval logic. CCM traverses the graph:

```
-- Security groups with SSH or RDP open to internet
FIND aws_security_group AS sg
  THAT ALLOWS AS rule internet
  WHERE
    (rule.fromPort <= 22 AND rule.toPort >= 22)
    OR
    (rule.fromPort <= 3389 AND rule.toPort >= 3389)
RETURN sg.name, sg.groupId, sg.region
```



WHY THIS MATTERS

Large organizations can have thousands of security groups across regions and accounts. CCM catches open-port misconfigurations simultaneously across all of them — not just a sampled subset.

WHAT SQL CAN'T DO

A relational database can't express the relationship between a security group, a rule edge, and the internet as a single graph traversal. In J1QL, it's one query.

CONTROL EXAMPLE · 3 OF 3

Finding VPCs Without Flow Logging

VPC Flow Logs are the primary record of traffic flows within and across a Virtual Private Cloud. Without them, security teams cannot detect port scanning, anomalous east-west traffic, or communications with a command-and-control server — a major blind spot in network telemetry.

Many frameworks — including SOC 2, ISO 27001, and HIPAA — expect organizations to retain evidence showing who had network access to sensitive systems. In an AWS VPC, enabling flow logging is the main way to meet that expectation. In SQL, you'd need a LEFT JOIN with a NULL check for each VPC. In J1QL, the query is a single keyword:

```
-- Find VPCs that lack flow logging (negative test)
FIND aws_vpc
  THAT !HAS aws_flow_log
RETURN
  aws_vpc.name,
  aws_vpc.vpcId,
  aws_vpc.region
```

WHY THIS MATTERS

Without flow logging, security teams cannot detect lateral movement, data exfiltration, or unauthorized access within the VPC. CCM continuously monitors for this gap — ensuring it's never open for long.

RELEVANT FRAMEWORKS

SOC 2 | ISO 27001 | HIPAA

KEY CAPABILITIES

Three Pillars of Modern CCM

An industry-leading CCM solution must excel in three core areas. Together, they define what separates genuine continuous monitoring from periodic check-box compliance.

#1

Flexible, Granular Controls

Defining controls as code — using a precise, composable query language like J1QL — enables unlimited flexibility. There are no vendor limits on which controls can be tested. If you can describe it in a query, you can encode it as a control, including complex contextual policies like "all internet-facing workloads processing PII must enforce MFA and logging."

Predefined framework controls | Custom policy controls | Complex, contextual policies

PROACTIVE RISK REDUCTION: THREAT INTELLIGENCE-INFORMED CONTROLS

Teams can identify specific attacker TTPs from their threat intelligence feed, translate them into concrete conditions, and test for each with J1QL. For example:

```
-- AWS admin accounts without MFA (a common attacker target)
FIND aws_iam_user WITH admin = true AND mfaEnabled != true
```

#2

Extensive Automation

When tests run automatically on a regular cadence against your environment, the benefits compound rapidly:

- No need for periodic manual testing
- Audit prep time shrinks from weeks to hours
- Consistent, ongoing validation measures current reality
- Security teams shift focus to threat response and high-value analysis
- Automated alerts assign remediation responsibility immediately

CLOSING THE LOOP

Integration with ticketing systems closes the detection-to-remediation loop, ensuring alerts translate directly into assigned, tracked work items.

#3

AI-Readiness

Bi-directional integration between a CCM solution and a Model Context Protocol (MCP) server makes it possible to retrieve information and build queries using natural language requests — making even the most complex control queries simple to run.

AI CONTROLS AUTHOR

Takes an entire written control catalog — policies, frameworks, requirements — and transforms it into a set of suggested controls with J1QL queries ready to run against a live asset graph.

IMPACT

By eliminating the lag time inherent in manual query creation, the AI Controls Author further reduces cyber risk and accelerates framework alignment.

CONCLUSION

The Future Is Continuous

CCM changes the role that security controls play in the enterprise. Instead of being periodically checked for audit purposes, they're continuously tested — validating that infrastructure aligns with desired security posture at all times.

As organizations adopt more cloud services, AI-driven workflows, and complex dynamic technologies, continuous validation becomes foundational. Compliance becomes a natural outcome of strong security practice — not a periodic scramble.

JUPITERONE CCM GIVES YOU

- A cyber asset graph continuously ingesting 200+ sources — cloud, SaaS, identity, endpoints, and application security tools
- Control tests that run against what's *actually* happening in the environment — not what your policies say should be happening
- Complex, multi-domain control testing across diverse, dynamic environments — exactly where most real-world risk resides today

Schedule a demo →