JupiterOne

Continuous Controls Monitoring in the Age of Ephemeral Infrastructure

Introduction

With the adoption of 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—both to reduce real-world cyber risk and for regulatory compliance—were designed for largely static environments. They simply can't keep pace with the current rate of change.

The most common legacy model for monitoring controls consisted of periodic manual checks, carried out on a quarterly or annual basis, or right before each audit. This strategy worked back when most infrastructure changes were made during infrequently-scheduled maintenance windows or occasional hardware refresh cycles. In yesterday's data center-based environments, security and compliance teams had to keep track of servers, networking equipment, a relatively small number of applications, employee identities, and not much else.

That's no longer the case. Modern, cloud-first enterprises incorporate thousands of different types of assets, resources, cloud workloads, and APIs, changing so quickly that a manual asset inventory is likely to produce results that are stale before it's even complete.

Traditional controls validation offered only a moment-in-time snapshot of the organization's security and compliance posture. Putting together this "snapshot" reactively—in preparation for an audit—inevitably left security teams exhausted, scrambling, and needing to work overtime. And as soon as the audit was complete, control drift would begin again.

Continuous controls monitoring (CCM) instead enables proactive, real-time validation of an enterprise's security controls. This ongoing assurance is the foundation of a robust security posture.

Once CCM has been implemented, compliance becomes a natural by-product of resilience, and audits will be simple, easy and (maybe) even enjoyable.

This eBook will explore the architectural approach engineers need to take in order to achieve successful CCM at enterprise scale. We'll also discuss controls-as-code, and explain why it is the future of security posture validation, especially in a cloud-native and AI-powered world.

What Is Continuous Controls Monitoring (CCM)?

Continuous Controls Monitoring is the process of automatically and continuously evaluating the effectiveness of security controls across an organization's infrastructure. It ensures that controls designed to mitigate risks, protect data, and maintain compliance with regulatory requirements are functioning as intended, and it delivers this assurance in real time.

CCM provides ongoing visibility into how well security measures are protecting the enterprise against threats. By identifying misconfigurations, control failures, and gaps, CCM makes it possible to correct them before an incident takes place.

CCM is profoundly different from legacy approaches to validating security controls. Here's how:

	CCM	Legacy approach
Test frequency	Continuous: controls are tested automatically and often	Periodic: tests typically performed right before audits
Evidence-gathering	Automatic	Manual: data is collected by hand, stored in spreadsheets or governance, risk, and compliance (GRC) tools, and emailed to auditors
Test coverage	Comprehensive: tests are run across the entire environment	Limited: only a sample subset of systems are tested
Ease of testing	Simple: Query-based control testing is easy to run, but also flexible and easy to customize	Labor-intensive: Control testing involves manual scripting, which takes effort
Approach	Proactive: Control drift is prevented, and evidence is available at all times	Reactive: Organizations test for controls in preparation for an audit and allow drift to occur afterwards; the next time around, there's more work to do
Results	Real-time insights, delivered within a detailed, drill-down dashboard	Static audit reports or spreadsheets

How Modern CCM Works

Legacy controls testing was largely manual: testers would inspect evidence by reviewing screenshots, reports, and user interfaces (UIs), or by watching an employee perform the process subject to the control. Because it's impossible to cover all aspects of an environment this way, evidence was selected through sampling. Testers would select a set of transactions, tickets, or infrastructure changes and test whether the control in question was applied correctly in each of those instances. The hope was that the sample would be representative, but there was no guarantee of this.

In modern CCM, by contrast, ***every control is tested.***

For each control, a discrete, deterministic test is defined (or multiple tests can be used), which returns a result validating that the control is present or absent. These tests can be negative or positive:

A **negative test** searches for violations; the control is in place across the entire environment if zero results are returned.

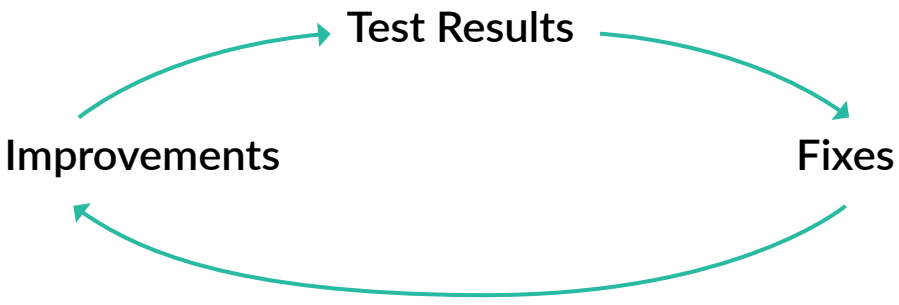
A **positive test** verifies that the tested-for configuration is in place; the desired finding is that it is, everywhere.

Each test is performed by querying the organization's **asset graph**, which represents all the cyber assets in the enterprise technology ecosystem and documents all of the relationships between those assets.

Tests run on automated schedules. Routine testing can be conducted daily, with additional tests triggered whenever security or compliance teams want to run them—for instance, when a change is known to have taken place. This way, control validation remains current.

The CCM dashboard displays the current status of each control (e.g., compliant, non-compliant), based on aggregated test results. Automated alerts let the person responsible for remediating the misconfiguration or control gap know what needs to be changed.

Modern CCM sets in motion an ongoing, automated process, establishing a feedback loop between the state of the enterprise's infrastructure and its security and compliance posture.



The Graph Database Backbone

What sets an industry-leading CCM solution apart is the database backend that powers it: a graph database. Conventional GRC tools and configuration management databases (CMDBs) are built on relational databases with rigid schemas. Relational databases cannot model complex relationships between assets and controls at scale.

By contrast, graph databases are designed to both store entities and illustrate the relationships between them. Instead of static rows and columns, a graph database is comprised of a dynamic entity-relationship map, where both entities and relationships can be described in multiple ways, incorporating rich object properties and metadata.

This model is a natural fit for cybersecurity use cases since the attack surface itself is a complex web of assets, and understanding cyber risk requires understanding the relationships and interdependencies between these assets.

A graph database can:

- Model complex technology environments accurately
- Support queries across millions of interrelated data points
- Connect the dots between cyber assets, relationships, and risks

Because of these capabilities, a CCM solution build atop a graph database will be able to:

- Scale massive amounts of enterprise data, including every relationship between users, devices, apps, data, cloud resources, and other attributes.
- Answer granular questions based on the most detailed data model in existence, across the entire environment.
- Test extremely complex controls, even across multi-cloud environments.

A graph database has a highly flexible schema, so it can support CCM that is highly configurable, interoperable, and extensible. A traditional database can't handle the extensive integrations that a graph database can support. It also won't be able to handle rapid growth and changes in schema, even though modern technology environments are fluid.

Let's take a look at what the graph database backend makes possible.

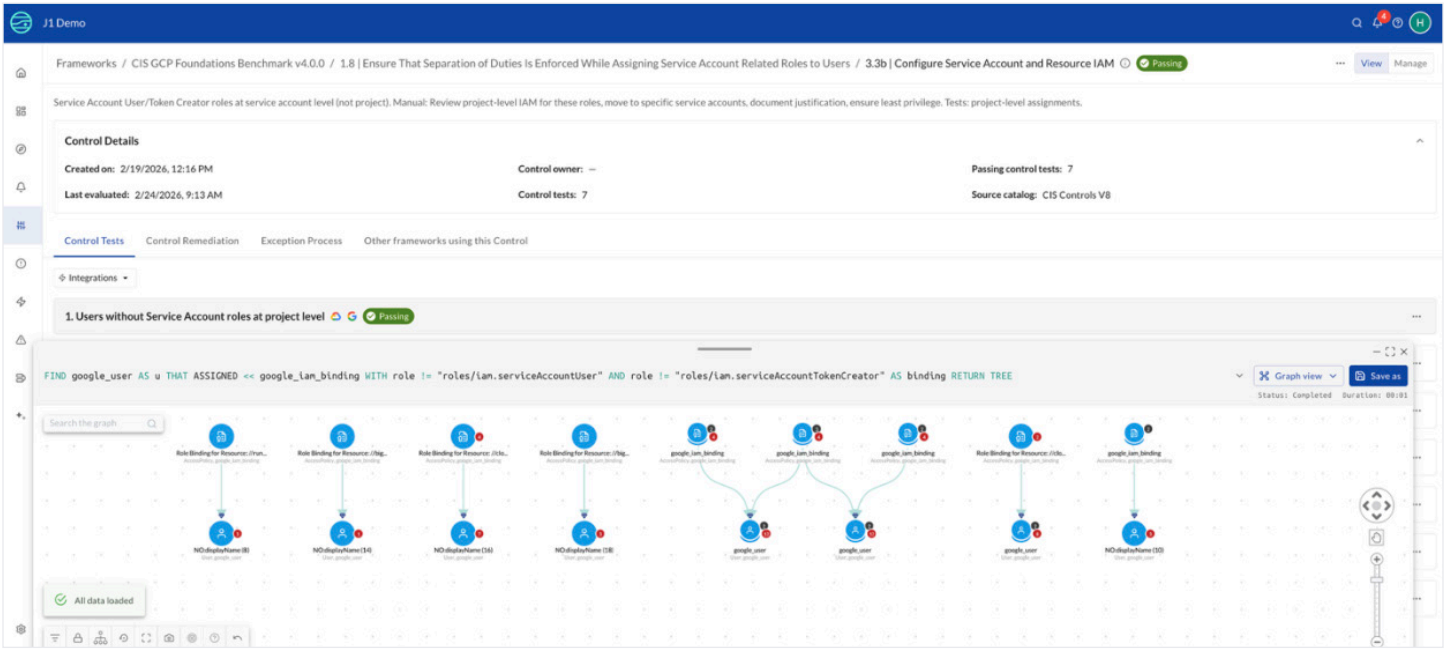
Finding a Segregation of Duties Violation in GCP

Privilege-related risks aren't always obvious. Sometimes it's the combination of permissions that's dangerous.

In Google Cloud Platform (GCP), it's possible to grant the same user both a Service Account Administrator role and a Service Account User role, but this toxic combination is risky. An attacker who gained access to that account could easily create additional backdoor service accounts, mint keys, and maintain persistent access even if credentials are rotated. This segregation of duties (SoD) violation is a common audit finding, appearing in multiple regulatory and industry-standard frameworks including the Sarbanes-Oxley Act (SOX), ISO 27001, the NIST Cybersecurity Framework (CSF), and CIS GCP Foundations Benchmark v4.

It's difficult to find this SoD violation through manual controls testing. In an SQL database search, you'd need to cross-reference evaluations of two different policies to find their intersection. And this process has to be repeated for each individual failure.

CCM not only automates the test for this control, but also demonstrates—with a single query—that the SoD violation is absent across the entire environment.



Finding AWS Security Groups with Open Ports

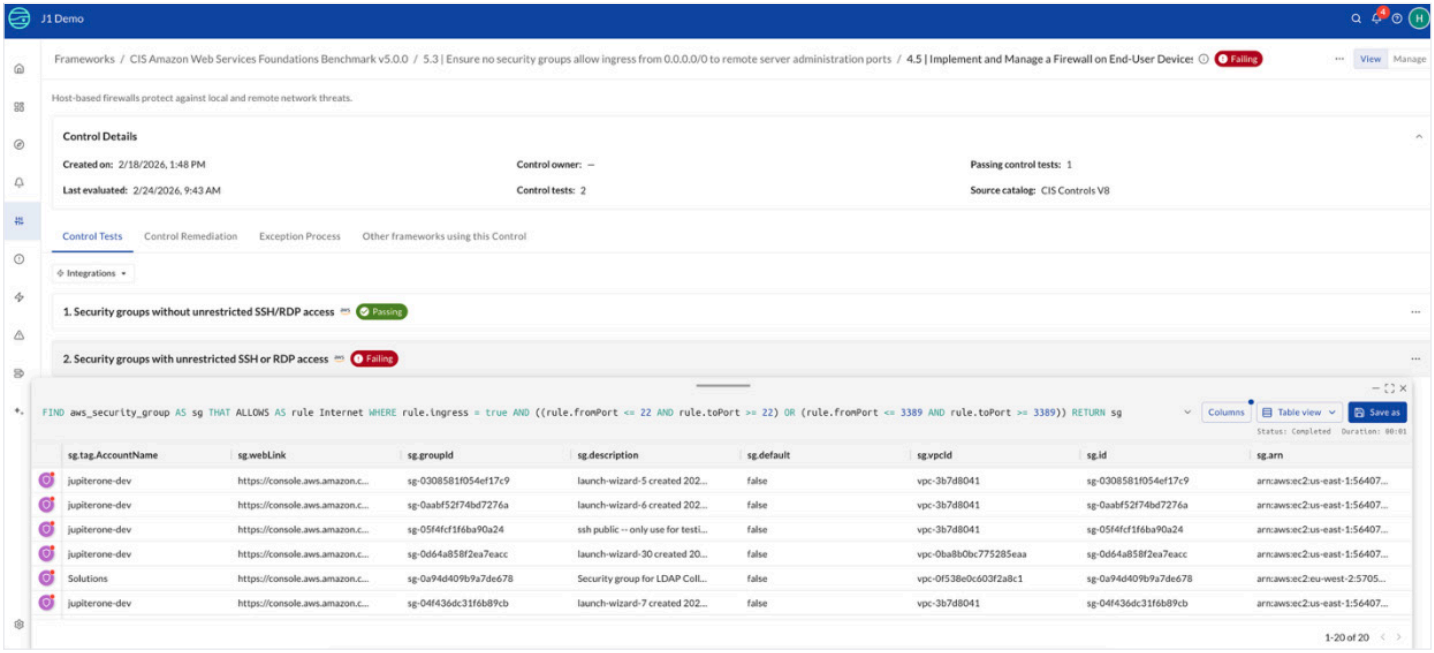
Small, temporary changes to AWS security groups can create major vulnerabilities if they're not reversed promptly.

When configuring security groups on AWS, it's a best practice to limit inbound access to ports 22 (SSH) or 3389 (RDP) to the specific IP addresses or ranges that require it. However, leaving these ports open is a common misconfiguration that's often found during cloud security reviews and audits. An engineer might open these ports while debugging software, or to give a vendor temporary access, but then forget to revert back to the initial settings. Or teams might re-use an overly permissive security group template across many instances, widely propagating the misconfiguration across their AWS environment.

This misconfiguration exposes a wide range of ports to any IP address on the open internet. Any attacker able to gain access—through brute force or with stolen credentials—would then have remote administrative privileges, enabling access to locally-stored data, cached keys and credentials, and other resources that enable lateral movement.

It can be difficult to detect all AWS security groups with unrestricted SSH/RDP access with a manual search, since large organizations can have thousands of security groups across regions and business units. And regex-based scanners will miss this port-interval logic.

With a CCM solution with a graph database backend, it's simple to run a query traversing the relationship between a security group, a rule edge, and the internet. This kind of relationship cannot be expressed in a traditional database.



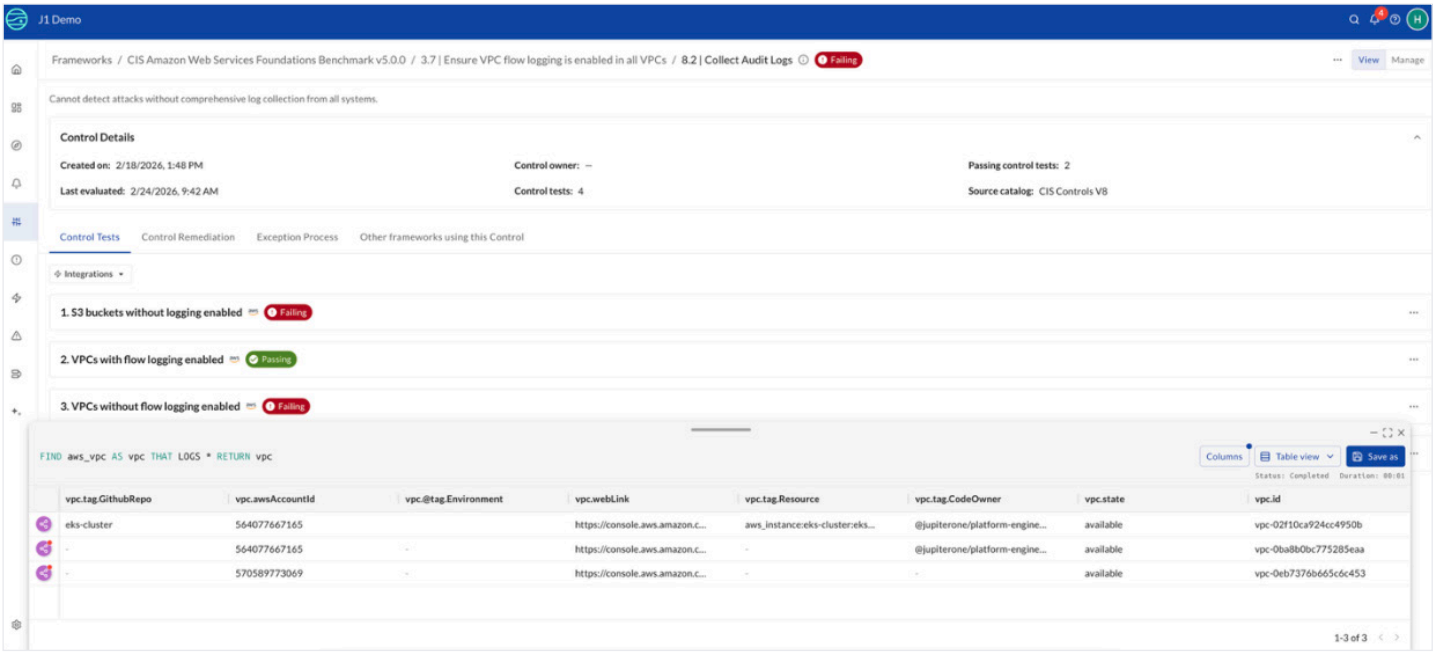
Finding VPCs Without Flow Logging

If you aren't recording data on the traffic flows within your Virtual Private Cloud, there's a major visibility gap that attackers won't hesitate to exploit.

Virtual Private Cloud (VPC) Flow Logs serve as a primary record of the traffic flows within and across a VPC. Without them, it's much harder to detect port scanning, anomalous east-west traffic, or communications with a command-and-control server. Many security and regulatory 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. Without it, security teams cannot detect lateral movement, data exfiltration, or unauthorized access within the VPC environment—a major blind spot in network telemetry.

Discovering that VPC Flow Logs are not enabled is a common security assessment and audit finding, but it's challenging to check for this misconfiguration in a relational database. In SQL, you'd need to combine a LEFT JOIN with a NULL check.

In J1QL, the query is a single keyword.



Key Capabilities to Look for in a CCM Solution

There are three core capabilities that an industry-leading CCM solution should have. We call these the three pillars of modern CCM.

They are:

- Flexible, Granular Controls
- Extensive Automation, and
- AI Readiness

Let's take a closer look into each area.

#1: Flexible, Granular Controls

While some enterprises are required to adhere to industry-standard regulatory frameworks such as SOC 2 or DORA, others may have built out their own organization-specific framework, or they may want to monitor additional controls to go beyond the scope of regulatory requirements. A modern CCM solution should make it simple to monitor any control in any part of the environment.

Defining **controls as code**, using a domain-specific, precise, composable query language such as JupiterOne Query Language (J1QL), enables unlimited flexibility. With this approach, there are no vendor-determined limits on which controls can be tested.

If you can describe it in a query, you can encode it as a control.

Because each control test is a query run against the entire organization's cyber asset graph, security and compliance teams can ***ask any question*** about their enterprise's asset collection and get an accurate answer. And they can turn those questions into automated control tests that run continuously.

Teams can choose from predefined controls based on commonly-applied frameworks or create their own to match industry-specific regulations, company-specific policies, or security practices they're interested in exploring.

Since the cyber asset graph models complex relationships between entities, CCM based on this model can describe and validate ***complex, contextual policies***, such as "all internet-facing workloads processing PII must enforce MFA and logging."

Proactive Risk Reduction: Building Threat Intelligence-Informed Controls

With a modern CCM solution, enterprises can establish threat intelligence-aligned controls and monitor them continuously. Teams would start by identifying specific attacker behaviors or tactics, techniques, and procedures (TTP) within their threat intelligence feed, and then translate these into concrete conditions (such as “exposed Kubernetes dashboard” or “AWS admin account lacking MFA”).

They can test for each concrete condition using the folllowing J1QL query, which will reveal any at-risk entities in the environment:

```
FIND aws_iam_user WITH admin = true AND mfaEnabled != true
```

Users can group these threat intelligence-driven controls into a custom control set or map them onto existing frameworks. The CCM solution’s dashboard will show pass/fail states, compliance percentages, and trends over time for each of these controls. As threat intelligence evolves, teams can more J1QL control queries, initiating automated testing for each additional threat pattern.

#2: Extensive Automation

Want to realize time and labor savings? Keep your organization in a state of continuous audit-readiness and eliminate pre-audit scrambles? The key CCM capability is automated controls testing.

When tests are run automatically, on a regular cadence, against your environment:

- There’s no longer any need for periodic manual testing.
- Control validation is quick, easy, and efficient.
- Audit prep time shrinks from weeks to hours.
- Security teams can shift their focus from manual testing and status updates to threat response, policy improvements, and high-value analysis.

Automation ensures that there’s consistent, ongoing control validation—and that it measures the current reality.

CCM should also generate alerts when controls are not adherent and assign responsibility for remediation. For this, integration with ticketing systems is essential, closing the detection-to-remediation loop.

#3: AI-Readiness

Bi-directional integration between an CCM solution and a model context protocol (MCP) server makes it possible to interact with the CCM solution—to retrieve information and build queries—using natural language requests. This capability makes it very simple to run extremely complex queries.

The AI Controls Author

The AI Controls Author goes one step beyond natural-language query creation. It takes an entire written control catalog (including policies, frameworks, requirements, and more) and transforms it into a set of suggested controls. It can also create the J1QL queries needed to test these controls against a live asset graph.

This dramatically reduces the amount of time it takes to go from a control framework to a set of continuously-tested technical controls, making it possible to keep controls fully aligned with today's threats and the latest version of all relevant frameworks. By eliminating the lag time inherent in manual query creation, the AI Controls Author further reduces cyber risk.

Conclusion

CCM changes the role that security controls play in the enterprise. Instead of being something that's periodically checked for audit and compliance purposes, they're continuously tested, validating that the organization's infrastructure is in line with its desired security posture. Security and compliance teams gain benefits from this that extend far beyond faster audit preparation or simplified reporting—though the advantages in these areas are substantial. But teams also gain ongoing assurance that they'll be able to detect control drift quickly, correct misconfigurations early, and reliably understand their organization's real-world risks at all times.

As organizations continue adopting cloud services, AI-driven workflows, and increasingly complex and dynamic technologies, these capabilities become foundational. Continuous validation ensures that security controls evolve alongside the environment they protect. In this model, compliance becomes a natural outcome of strong security practices rather than the primary objective. Organizations move beyond point-in-time compliance snapshots toward a future-focused approach where security posture is constantly being measured, understood, and improved.

Built from the start with an entity-relationship graph as its foundational data model, JupiterOne's cyber asset graph can fully capture every aspect of today's complex, cloud- and SaaS-based digital environments. JupiterOne continuously ingests and normalizes data from more than 200 sources (including cloud, SaaS, identity, and application security tools as well as endpoint devices) so that control tests run against what's actually happening in the environment, not just what your policies say should be happening. The cyber asset graph models how users, roles, devices, apps, and networks connect with one another, making it possible to define and test complex, multi-domain controls across diverse, dynamic environments—exactly where most real-world risk resides today.