# Monitoring and Auditing in AWS

Booklet

# Foreword

Cloud observability has never been more consequential. As organizations move critical workloads to AWS at scale, the ability to detect threats in real time, maintain audit trails, and prove continuous compliance has become a baseline expectation — not an advanced capability.

The stakes are higher, the attack surface is broader, and the regulatory landscape more demanding than at any previous point in the cloud era. The average enterprise now runs workloads across dozens of AWS accounts, spanning multiple regions, hundreds of services, and thousands of individual resources — all generating telemetry that must be collected, stored, analyzed, and acted upon.

This manuscript is written as a practitioner's companion: equal parts conceptual grounding and actionable guidance. Every chapter reflects the service state as of April 2026, incorporating the major announcements from AWS re:Invent 2025 and the Q1 2026 release cycle. Readers will find new material on CloudWatch's unified data store (December 2025), CloudTrail Insights for data events (November 2025), the generally available Security Hub (December 2025), GuardDuty Extended Threat Detection for EC2 and ECS (December 2025), Amazon Inspector's agentless Windows scanning (March 2026), and the transition of X-Ray to OpenTelemetry (February 2026) — among many others.

The book is organized in three arcs. Chapters 1–3 lay the foundation: what monitoring and auditing mean in the cloud, the anatomy of the AWS observability platform, and how to structure telemetry from the ground up. Chapters 4–7 dive deep into the core services — CloudWatch, CloudTrail, AWS Config, and the security-detection layer (GuardDuty, Security Hub, Inspector, Macie). Chapters 8–10 address production concerns: distributed tracing and application observability, multi-account architectures, compliance automation, and operational maturity.

Whether you are a solutions architect designing the observability layer for a new landing zone, a security engineer hardening a regulated workload, or a developer troubleshooting a distributed system, this guide is structured to be useful at every level of experience. Code examples throughout the book use Terraform (v1.7+), AWS CLI (v2), and Python, with YAML for configuration snippets where appropriate.

A note on scope: this book focuses exclusively on AWS-native tooling. Third-party SIEM platforms, APM products, and observability SaaS tools (Datadog, Splunk, Dynatrace, etc.) are referenced where relevant for integration context, but their configuration is outside the scope of this guide.

# Table of Contents

## Chapter 10 — Compliance, Best Practices & Operational Maturity

CHAPTER 1

# Foundations of Cloud Observability

Understanding what you need to see — and why it matters

## 1.1 What Is Observability?

Observability is the degree to which the internal state of a system can be inferred from its external outputs. The term originates from control theory, where a system is called 'observable' if knowledge of all its outputs over a finite time interval is sufficient to determine its internal state. In modern software engineering, the concept has been adapted to describe a practice: the deliberate instrumentation of systems so that engineers can understand their behavior from the outside, without requiring modifications or restarts.

The distinction between observability and traditional monitoring is subtle but important. Monitoring asks a predefined set of questions — questions you anticipated when you set up your dashboards and alarms. If your CPU alarm is set to fire at 90%, you will know when CPU hits 90%, but you will not know why a request took 8 seconds last Tuesday without going back and examining raw logs. Observability is the property that lets you investigate conditions you never anticipated — to explore the state space of a complex distributed system after something has already gone wrong.

AWS has progressively evolved its platform from a collection of individual monitoring tools toward an integrated observability platform. The 2025–2026 release cycle in particular marked a step change: the CloudWatch unified data store (December 2025) collapses the need for separate pipelines for operational, security, and compliance data. The auto-enablement rules (Q1 2026) move toward a 'zero-touch telemetry' model where any new resource automatically begins emitting the right signals to the right destination.

## 1.2 Monitoring vs. Auditing vs. Logging

These three terms are frequently conflated but serve distinct purposes in an AWS environment. Understanding their boundaries is critical for designing the right data pipeline and for satisfying compliance requirements.

Monitoring is the practice of watching for predefined conditions — typically operational or performance conditions — and alerting when those conditions are violated. Monitoring is inherently real-time (or near-real-time) and action-oriented. Its primary consumers are operations teams. The key question monitoring answers is: 'Is something wrong right now?'

Logging is the practice of recording detailed event sequences for later analysis. Log data is richer than metric data — it preserves the full context of what happened — but it is more expensive to store and

slower to query at scale. Logs support both operational debugging (why did this Lambda function fail?) and security forensics (what did this IAM user do?). The key question logging answers is: 'What exactly happened, and in what order?'

Auditing is the practice of maintaining an immutable, tamper-evident record of actions for governance, compliance, and accountability purposes. Audit data must be reliable — you cannot allow it to be deleted, modified, or selectively captured. The primary consumers of audit data are compliance teams, security investigators, and external auditors. The key question auditing answers is: 'Can you prove what happened, when, and by whom?'

Tracing has emerged as a fourth distinct discipline, focused on following the path of a single transaction through a distributed system. Tracing answers the question: 'Why was this specific request slow, and which service was responsible?'

| Concept | Primary Purpose | Key AWS Services | Retention Focus |
|---|---|---|---|
| Monitoring | Detect operational/performance anomalies in near-real-time | CloudWatch Metrics, Alarms, Dashboards | Short (days to weeks) |
| Logging | Record event sequences for debugging and forensics | CloudWatch Logs, S3, CloudTrail Lake | Medium-to-long (months to years) |
| Auditing | Prove what happened, who acted, when — for compliance | CloudTrail, AWS Config, Audit Manager | Long (years); tamper-evident |
| Tracing | Follow a request end-to-end through distributed components | X-Ray, ADOT, Transaction Search | Short (days); sampled |

Table 1.1 — Core observability concepts and their AWS service mapping.

## 1.3 The Three Pillars: Metrics, Logs, Traces

The 'three pillars of observability' framework, popularized by Charity Majors and codified in the OpenTelemetry specification, provides a mental model for structuring telemetry collection. Each pillar has different characteristics, costs, and use cases.

### Metrics

Metrics are numeric measurements sampled at regular intervals. They are cheap to store, fast to query, and well-suited for alerting and trend analysis. Every AWS service emits default metrics to Amazon CloudWatch. EC2 emits CPU utilization, network I/O, and disk metrics every five minutes by default; with detailed monitoring enabled, this becomes one-minute granularity. Custom metrics can be published via the PutMetricData API, the CloudWatch agent, or the Embedded Metrics Format (EMF) within Lambda and ECS workloads.

The fundamental limitation of metrics is their inability to explain causality. A spike in p99 latency tells you something is wrong; it does not tell you which service is at fault, which customer is affected, or what code path was executing. Metrics answer 'what' but rarely 'why.'

### Logs

Logs are discrete, timestamped records of events. They are richer than metrics but more expensive to store and query at scale. CloudWatch Logs accepts structured (JSON) and unstructured text logs from virtually any source: Lambda functions, EC2 instances, ECS containers, API Gateway, VPC flow logs, ALB access logs, CloudFront, and many others.

The Logs Insights query engine provides a SQL-like syntax for ad-hoc exploration. As of December 2025, CloudWatch's unified data store allows log data to be materialized into Amazon S3 Tables as Apache Iceberg files, enabling analysis with Athena, SageMaker Unified Studio, Amazon Redshift, and any Iceberg-compatible tool without additional ETL pipelines.

### Traces

Traces follow the path of a single request through all the services it touches, recording latency, errors, and context at each hop. AWS X-Ray has provided distributed tracing since 2016. In February 2026, the X-Ray SDKs and Daemon entered maintenance mode, and AWS now recommends the AWS Distro for OpenTelemetry (ADOT) as the primary instrumentation path. X-Ray continues to be the backend storage and visualization service.

The relationship between the three pillars is complementary and hierarchical in practice. You start with metrics to detect that something is wrong (alarm fires). You drill into logs to understand what happened in that service during the anomaly window. You then use traces to understand the full call chain and identify the root cause component. Each pillar fills a gap that the others cannot.

## 1.4 AWS Shared Responsibility Model

The AWS Shared Responsibility Model defines clear boundaries between what AWS monitors on your behalf and what you are responsible for monitoring yourself. A common misconception is that using a managed service means AWS monitors it for you — in fact, managed services reduce operational burden but do not eliminate your observability responsibilities.

AWS is responsible for monitoring the health of the underlying physical infrastructure (hardware, hypervisors, network fabric, data center facilities), the managed service control planes (RDS engine health, S3 storage systems, Lambda execution environment), and global service availability. AWS publishes service health on the AWS Service Health Dashboard.

You are responsible for monitoring your applications and workloads running on AWS, the configuration of AWS services (tracked by AWS Config), access patterns and user activity (recorded by CloudTrail), security findings within your accounts (surfaced by GuardDuty, Inspector, and Security Hub), compliance with regulatory standards applicable to your industry, and data quality and availability within your databases and storage services.

Note: Key Design Principle

The shared responsibility model does not reduce your monitoring burden — it changes its focus. You own the data plane. Every EC2 instance, container, Lambda function, and database query is your responsibility to instrument and monitor. Plan accordingly from day one.

## 1.5 Designing for Observability from Day One

Retrofitting observability onto an existing system is expensive, error-prone, and almost always incomplete. The highest-performing engineering organizations treat telemetry as a first-class feature requirement with its own acceptance criteria: a service is not 'done' until it emits the right metrics, writes structured logs, and propagates trace context.

In AWS, the key practices for building observability in from the start are:

- Enable CloudTrail organization-wide before any workloads are deployed — make it part of your AWS Organizations bootstrap.
- Define a log retention policy in AWS Config and enforce it with a Config Rule on all CloudWatch log groups.
- Codify CloudWatch alarms, dashboards, and log groups in Terraform or CDK alongside the resources they monitor — never configure them manually in the console.
- Require ADOT instrumentation as part of your application platform standards — create a template or golden path that makes it easy for developers to add tracing.
- Enable GuardDuty, Security Hub, and AWS Config across all accounts from day one using AWS Organizations delegated administration.
- Establish a tag governance policy — consistent resource tagging propagates into metric dimensions and Config evaluations, enabling filtering and attribution at scale.
- Set CloudWatch Log Group retention periods at creation — the default is 'never expire', which results in unbounded storage costs over time.

## 1.6 The Cost of Poor Observability

Poor observability is not just a technical inconvenience — it has measurable business consequences. When a production incident occurs without adequate telemetry, engineers spend hours gathering data that should have been collected automatically. Mean Time to Detection (MTTD) stretches from minutes to hours. Mean Time to Resolution (MTTR) follows the same pattern.

From a compliance perspective, the costs are even higher. Regulators increasingly expect organizations to demonstrate continuous monitoring, not just produce logs on demand during an audit. PCI DSS v4.0 (effective April 2025 for organizations on extension) requires real-time alerting for specific event types and evidence that alerts are acted on. HIPAA enforcement actions routinely cite inadequate access logging as a contributing factor. SOC 2 Type II audits examine whether monitoring controls operated effectively throughout the audit period — not just whether they were configured.

The AWS Well-Architected Framework's Security Pillar explicitly calls out 'enabling traceability' as a foundational security design principle — every action should be logged, and every log should be analyzable.

CHAPTER 2

# The AWS Observability Ecosystem

A map of every service and how they connect

## 2.1 Service Landscape Overview

AWS offers more than a dozen services that collectively address monitoring, logging, auditing, and security detection. Understanding the full landscape before choosing tools prevents point-solution sprawl and overlapping data collection costs.

| Category | Service | Primary Function | Launch / Major Update |
|---|---|---|---|
| Metrics & Alarms | Amazon CloudWatch | Collect, visualize, alert on metrics | Unified data store — Dec 2025 |
| Logging | CloudWatch Logs | Managed log aggregation and query | Log Centralization — Mar 2026 |
| API Audit | AWS CloudTrail | Record all AWS API activity | Insights for data events — Nov 2025 |
| Config Compliance | AWS Config | Track resource configuration changes | Additional regions — May 2025 |
| Threat Detection | Amazon GuardDuty | ML-based threat detection | ECS/EC2 Extended Detection — Dec 2025 |
| Security Posture | AWS Security Hub | Aggregate and correlate security findings | GA Dec 2025; GovCloud Mar 2026 |
| Vulnerability Mgmt | Amazon Inspector | CVE scanning for EC2, ECR, Lambda | Agentless Windows — Mar 2026 |
| Data Classification | Amazon Macie | Sensitive data discovery in S3 | Ongoing enhancements |
| Tracing | AWS X-Ray | Distributed request tracing | OTel transition; SDKs maintenance Feb 2026 |
| Network | VPC Flow Logs | Network traffic metadata capture | Org-wide auto-enable — Aug 2025 |
| App Firewall | AWS WAF | Web application filtering + logging | Free vended logs — Sep 2025 |
| Audit Reporting | AWS Audit Manager | Evidence collection for audits | Closing to new customers Apr 30, 2026 |
| AI Security | AWS Security Agent | Automated app security review + pen-testing | Preview — Dec 2025 |

| Category | Service | Primary Function | Launch / Major Update |
|---|---|---|---|
| Data Streaming | Amazon Kinesis Firehose | Real-time log/event delivery to destinations | Ongoing |
| SIEM Integration | Amazon Security Lake | OCSF-normalized security data lake | Ongoing |

Table 2.1 — AWS observability and security service landscape as of April 2026.

## 2.2 How Services Interconnect

Each service generates telemetry that feeds other services, creating a deeply integrated detection and response fabric. The following data flows are the most important to understand:

CloudTrail to CloudWatch Logs: As of December 2025, CloudTrail events can be ingested directly into CloudWatch Logs via service-linked channels (SLCs), without requiring a separate trail configuration. This enables CloudWatch metric filters and alarms to fire on specific API calls within seconds of their occurrence. The SLC approach also provides termination protection — CloudWatch will refuse to stop collecting events if doing so would violate your retention policy.

GuardDuty / Inspector / Macie to Security Hub: All three services publish findings to Security Hub in the normalized AWS Security Finding Format (ASFF). Security Hub v2 (GA December 2025) correlates signals across sources, applies a unified severity scoring model, and surfaces attack paths through its graph visualization engine. A single compromised credential can generate findings in GuardDuty (behavioral anomaly), Inspector (vulnerability enabling lateral movement), and Macie (data exfiltration) that Security Hub presents as a unified attack sequence.

Security Hub / GuardDuty to EventBridge to Lambda/SSM: The automation backbone. EventBridge rules pattern-match on ASFF findings and invoke Lambda functions or Systems Manager Automation documents to isolate instances, revoke credentials, open ITSM tickets, or run forensic playbooks automatically — often within 60 seconds of initial detection.

AWS Config to Audit Manager: Config rule evaluations become compliance evidence artifacts in Audit Manager assessments, mapped to SOC 2, PCI DSS, or HIPAA controls. For organizations running recurring SOC 2 Type II audits, this automated evidence collection pipeline can reduce audit preparation time from weeks to days.

CloudWatch to Apache Iceberg (S3 Tables): The December 2025 unified data store allows CloudWatch logs to be materialized into Iceberg tables in managed S3, queryable by Athena, Amazon Redshift, or SageMaker Unified Studio. This eliminates the traditional ETL pipeline from CloudWatch to S3 for analytics use cases.

VPC Flow Logs to CloudWatch Insights: Flow logs delivered to CloudWatch Logs can be queried with Logs Insights using a custom field parser for the flow log format, or with native dashboards built into the CloudWatch console. The December 2025 unified data store enables cross-joining flow logs with CloudTrail events and GuardDuty findings in a single Athena query.

## 2.3 Pricing and Cost Management

Observability costs are one of the most underestimated line items in an AWS bill. Without active management, it is common for CloudWatch Logs, CloudTrail data events, and GuardDuty to collectively account for 10–25% of total AWS spend in security-conscious organizations.

| Service | Pricing Model | Key Cost Driver | Optimization |
|---|---|---|---|
| CloudWatch Metrics | Per metric per month after free tier | High cardinality custom metrics | Use EMF instead of PutMetricData |
| CloudWatch Logs | Per GB ingested + stored + queried | Verbose unstructured logs | Use S3 for high-volume, low-query |
| CloudTrail | Management events: 1 free copy; data events: $2/100k | All-S3 data events | Scope to sensitive buckets only |
| GuardDuty | Per GB of analyzed telemetry (tiered) | Large VPC flow log volumes | Review and tune protection plans |
| Security Hub | Per finding ingested (tiered) | High-volume low-value findings | Suppress with automation rules |
| Inspector | Per instance-month + per image scan | Large container image libraries | Scope to production ECR repos |
| X-Ray / ADOT | Per trace segment recorded | 100% sampling in production | Set sampling rate to 5% in prod |
| AWS Config | Per config item recorded + per rule eval | Frequent-change resource types | Use periodic eval for stable types |

Table 2.2 — Pricing model and cost optimization for observability services.

## 2.4 Data Retention and Lifecycle

Retention requirements are set by both operational needs and regulatory mandates. The two are often in tension: operations teams want data deleted quickly to reduce costs, while compliance teams need long retention periods to satisfy audit requirements. A tiered storage strategy satisfies both.

| Data Type | Hot (Immediate Access) | Warm (Low-Latency) | Cold (Archive) | Reg. Floor |
|---|---|---|---|---|
| CloudTrail management events | CloudTrail Lake: 90 days | S3 Standard: 1 year | S3 Glacier: 7 years | PCI: 1yr |
| CloudTrail data events | CloudWatch Logs: 30 days | S3 Standard: 90 days | S3 Glacier: 3 years | PCI: 1yr |
| VPC Flow Logs | CloudWatch Logs: 30 days | S3 Standard: 90 days | S3 Glacier: 1 year | HIPAA: 6yr |
| Application logs | CloudWatch Logs: 14 days | S3 Standard: 90 days | Delete or Glacier | Varies |

| Data Type | Hot (Immediate Access) | Warm (Low-Latency) | Cold (Archive) | Reg. Floor |
|---|---|---|---|---|
| GuardDuty findings | Security Hub: 90 days | S3 via export: 1 year | S3 Glacier: 7 years | SOC2: 90d |
| Config history | Config service: 90 days | S3 Standard: 1 year | S3 Glacier Deep Archive: 7yr | SOC2: 1yr |
| X-Ray traces | X-Ray service: 30 days | N/A | N/A | None |

Table 2.3 — Recommended data retention tiers for AWS observability data.

## 2.5 Choosing the Right Storage Tier

The choice of storage tier affects both cost and query latency. CloudWatch Logs is the most expensive storage option per GB but offers the fastest query time and native integration with CloudWatch Alarms and Insights. S3 Standard is significantly cheaper but requires Athena or a data lake query engine. S3 Glacier offers the lowest storage cost but requires hours to restore data.

The decision matrix is simple: data you need to alert on or query within minutes stays in CloudWatch Logs. Data you need to analyze within hours goes to S3 Standard. Data you need to retain for compliance but rarely access goes to S3 Glacier or Glacier Deep Archive.

The Apache Iceberg integration in CloudWatch's December 2025 unified data store changes this calculus slightly — data stored in S3 Tables (Iceberg format) can be queried with Athena in seconds rather than minutes, narrowing the query latency gap between CloudWatch Logs and S3 for analytical workloads.

CHAPTER 3

# Telemetry Architecture and Design Patterns

Structuring your data pipeline for scale, compliance, and cost

## 3.1 Log Routing and Aggregation

At scale, logs from dozens of AWS services and hundreds of applications need a coherent routing strategy before they arrive at a storage or query layer. Without a deliberate architecture, you end up with logs scattered across multiple CloudWatch log groups in multiple accounts with inconsistent naming, retention settings, and encryption — making forensic investigation and compliance reporting extremely difficult.

CloudWatch Log Subscriptions: A Kinesis Data Streams or Kinesis Firehose subscription on a log group routes matching events to a custom destination in near-real-time. This is the standard pattern for streaming logs to Amazon OpenSearch Service (formerly Elasticsearch) for full-text search, or to Splunk and other SIEM platforms via Firehose's HTTP endpoint destination.

CloudWatch Telemetry Config / Log Centralization: Launched at AWS re:Invent 2025 and enhanced in Q1 2026, CloudWatch centralization rules allow an organization to define collection policies at the organization, OU, account, or tag level. As of March 2026, data source targeting allows rules to be scoped to specific log source types — for example, collect only VPC Flow Logs from production-tagged VPCs in all accounts. This replaces hundreds of individually configured log subscriptions with a single declarative policy.

Amazon Data Firehose: Buffers and batches streaming data before delivery to S3, OpenSearch Service, Splunk, Datadog, or any HTTPS endpoint. Firehose supports format transformation (converting JSON to Parquet or ORC for efficient Athena querying), data compression, and dynamic partitioning (splitting data into S3 prefix paths by time or content field). This is the recommended path for high-volume log delivery to S3 where Athena will be the primary query engine.

AWS Lambda log subscriptions: For complex transformation or routing logic — enriching log events with metadata from a CMDB, routing different log types to different destinations based on content — a Lambda function subscription provides full flexibility at the cost of additional infrastructure to maintain.

## 3.2 Metric Collection Strategies

There are four primary paths for getting metrics into CloudWatch. The right choice depends on the source (managed service, EC2, container, Lambda), the cardinality of dimensions, and the required resolution.

- Default service metrics: Emitted automatically by every AWS managed service. No configuration required. Resolution is typically 1–5 minutes. Default metrics are the most cost-effective option and should always be the starting point before considering custom metrics.
- CloudWatch Agent: Installed on EC2 instances or ECS/EKS containers to collect OS-level metrics (memory, disk, custom process stats) and stream log files to CloudWatch Logs. The agent supports OTLP input, allowing it to receive OpenTelemetry metric exports from application instrumentation alongside OS metrics.
- Embedded Metrics Format (EMF): A JSON structure written to stdout by Lambda functions or ECS containers. CloudWatch automatically extracts metric dimensions and values from EMF payloads — no SDK calls required. EMF is the recommended approach for serverless workloads because it eliminates the latency overhead of synchronous PutMetricData calls during Lambda execution.
- PutMetricData API: Direct API calls for custom business metrics. Higher latency than EMF and expensive at scale (standard resolution metrics are priced per 1,000 PutMetricData data points). Use for business-level KPIs (orders per minute, checkout success rate) where the cardinality is low and the business value justifies the cost.

## 3.3 Trace Propagation

Distributed tracing requires a trace context (trace ID + span ID) to be propagated across every service boundary. If the context is lost at any hop — for example, at an SQS queue or a Lambda function invocation boundary — the trace chain breaks and you cannot reconstruct the full path.

OpenTelemetry defines two propagation formats: the W3C Trace Context format (traceparent header, the IETF standard) and the B3 format (used by Zipkin-compatible systems). AWS X-Ray uses its own format (X-Amzn-Trace-Id header) but supports both W3C and its native format since 2024.

For HTTP synchronous calls, context propagation is automatic when using ADOT auto-instrumentation. For asynchronous messaging (SQS, SNS, EventBridge), the trace context must be injected into message attributes by the producer and extracted by the consumer — ADOT SDK handles this automatically for supported services.

For Lambda functions, the runtime propagates the trace context from the invocation metadata automatically when X-Ray active tracing is enabled in the function configuration. For Step Functions, trace context flows through state machine executions and is propagated to Lambda and ECS tasks invoked within the state machine.

## 3.4 Multi-Region Telemetry

Most production AWS deployments span multiple regions for resilience or data residency reasons. The observability architecture must explicitly account for cross-region aggregation, or you will end up with siloed visibility that makes incident investigation much harder.

- CloudTrail multi-region trails (isMultiRegionTrail=true) write events from all regions to a single S3 bucket in the home region. This is the most important cross-region configuration — a single-region

- trail leaves a significant audit gap.
- CloudWatch cross-region dashboards can display metrics from any region in a single view, but CloudWatch Alarms can only reference metrics in the same region as the alarm. For cross-region alerting, use EventBridge with cross-region event buses.
- Security Hub supports cross-region aggregation with a designated aggregation region collecting findings from all linked regions. This should be configured on day one — retrofitting it later requires re-processing historical findings.
- GuardDuty must be enabled independently in each region, then linked to a delegated administrator account through AWS Organizations. GuardDuty findings are regional; the delegated admin sees all findings from all enabled regions.
- Amazon Inspector, Macie, and AWS Config all have similar per-region enablement models with cross-account aggregation via Organizations delegated administration.

## 3.5 Infrastructure as Code for Observability

Treating observability configuration as code is essential for repeatability, auditability, and avoiding configuration drift. When alarms, dashboards, log retention settings, and GuardDuty configuration exist only in the AWS console, they are at constant risk of being modified or deleted by well-intentioned operators.

The recommended approach is to co-locate observability resources with the application resources they monitor. Your Terraform module for an ECS service should include the CloudWatch log group with its retention setting, the metric alarms for task count and CPU, the dashboard widget definition, and the ADOT task definition extension. This makes observability an inherent property of the service rather than an afterthought.

```
# Terraform: compliance-grade CloudTrail with full controls
resource "aws_cloudtrail" "org_compliance" {
name = "org-compliance-trail"
s3_bucket_name = aws_s3_bucket.audit_logs.id
include_global_service_events = true
is_multi_region_trail = true # Critical - catches all regions
is_organization_trail = true # Covers all member accounts
enable_logging = true
enable_log_file_validation = true # SHA-256 tamper detection
kms_key_id = aws_kms_key.audit.arn
cloud_watch_logs_group_arn = "${aws_cloudwatch_log_group.audit.arn}:*"
cloud_watch_logs_role_arn = aws_iam_role.cloudtrail_cw.arn

event_selector {
read_write_type = "All"
include_management_events = true
```

```
data_resource {
type = "AWS::S3::Object"
values = ["arn:aws:s3:::sensitive-data-bucket/"]
}
data_resource {
type = "AWS::Lambda::Function"
values = ["arn:aws:lambda"] # All Lambda functions
}
}
tags = { Compliance = "SOC2,PCI-DSS,HIPAA", Environment = "production" }
}

resource "aws_cloudwatch_log_group" "audit" {
name = "/compliance/cloudtrail"
retention_in_days = 400 # >1 year satisfies PCI DSS requirement
kms_key_id = aws_kms_key.audit.arn
}
```

Code 3.1 — Terraform: compliance-grade organization CloudTrail.

## 3.6 Tag Strategy

Resource tagging is the foundational enabling capability for almost all advanced observability features at scale. Without consistent tags, you cannot scope CloudWatch alarms to 'all production databases', cannot create Security Hub rules that apply only to 'PCI in scope' resources, and cannot attribute costs or findings to specific teams or applications.

| Tag Key | Example Values | Used By |
|---|---|---|
| Environment | production, staging, development | Config rules, CloudWatch auto-enablement, cost allocation |
| Application | payments-api, auth-service | CloudWatch dashboards, X-Ray service map groupings |
| Owner | platform-team, security-team | SNS alarm routing, cost attribution |
| DataClass | pci-in-scope, hipaa-phi, public | Macie, GuardDuty, Security Hub scope filtering |
| CostCenter | cc-1234, cc-5678 | AWS Cost Explorer, Billing |
| ManagedBy | terraform, cdk, manual | Audit / drift detection |

Table 3.1 — Recommended resource tags and their observability use cases.

CHAPTER 4

# Amazon CloudWatch — Metrics, Logs & Alarms

The central nervous system of AWS observability

## 4.1 Metrics and Namespaces

CloudWatch organizes metrics by namespace, metric name, and a set of dimensions. A namespace is a container for related metrics — AWS services use namespaces like AWS/EC2, AWS/Lambda, and AWS/RDS. Custom metrics go into any namespace you choose (best practice: use a namespace like CompanyName/ServiceName). Dimensions are key-value pairs that uniquely identify a metric within a namespace.

CloudWatch stores metrics at tiered resolution as they age: 1-second and 1-minute data for 15 days, 5-minute data for 63 days, and 1-hour data for 455 days. This tiered storage model automatically reduces resolution as data ages, controlling storage costs without requiring explicit lifecycle configuration.

Metric Math: CloudWatch Metric Math allows you to compute new metrics from existing ones using mathematical expressions and functions. This is powerful for creating composite health indicators: you can compute error rate as errors/requests, or alert on a metric only when it is anomalous relative to another (e.g., latency is high when traffic volume is also high). Metric Math results can be used in alarms and dashboards without storing additional metrics.

High-resolution metrics: Custom metrics can be published at sub-minute resolution (as granular as 1 second) using the StorageResolution parameter in PutMetricData. This enables detection of sub-minute spikes in latency or error rate — important for high-frequency trading platforms or real-time gaming backends. High-resolution metrics are priced at a premium but offer significantly better anomaly detection for bursty workloads.

Cross-account and cross-region metrics: With CloudWatch Observability Access Manager (OAM), a monitoring account can display and alarm on metrics from all linked source accounts in a single view. This is essential for multi-account organizations and is covered in detail in Chapter 9.

## 4.2 CloudWatch Logs and Log Insights

CloudWatch Logs accepts structured and unstructured log events up to 256 KB in size. Events are organized into log groups (one per application or service) and log streams (one per instance or invocation). Log groups are the unit of retention configuration, encryption, and subscription.

Metric Filters: Metric filters extract metric values from log events in near-real-time. A filter pattern matches specific log lines (e.g., all events containing 'ERROR'), and CloudWatch increments or sets a metric value when a match occurs. This is how you convert unstructured application logs into alarming metrics without modifying your application code.

Logs Insights: A purpose-built query language for analyzing log data. Its syntax supports filter, stats, sort, limit, and parse (for extracting fields from unstructured text). Queries run against log groups selected at query time, and results are cached. Key functions include count(*), sum(field), avg(field), pct(field, 99) for percentiles, and ispresent(field) for checking field existence.

```
# Find the 10 IAM users with the most failed login attempts
fields @timestamp, userIdentity.userName, eventName, errorCode
| filter eventName = 'ConsoleLogin' and errorCode = 'Failed authentication'
| stats count(*) as failed_attempts by userIdentity.userName
| sort failed_attempts desc
| limit 10

# P99 Lambda cold start latency by function
filter @type = 'REPORT'
| parse @message 'Init Duration: * ms' as initDuration
| filter ispresent(initDuration)
| stats pct(initDuration, 99) as p99_coldstart by functionName
| sort p99_coldstart desc
```

Code 4.1 — CloudWatch Logs Insights: IAM failed logins and Lambda cold starts.

Contributor Insights: An often-overlooked CloudWatch feature that identifies the top contributors to a metric — for example, which source IP addresses are generating the most API errors, or which user agents account for the most WAF blocks. Contributor Insights uses a rule-based model to analyze high-cardinality log data and surfaces results in real-time dashboards without requiring explicit Logs Insights queries.

## 4.3 Alarms and Composite Alarms

CloudWatch Alarms watch a single metric (or metric math expression) and transition between OK, ALARM, and INSUFFICIENT_DATA states based on configurable thresholds. The threshold can be static (CPU > 80%) or anomaly-based (outside the predicted band). Actions on state transitions include SNS topic notification, Auto Scaling policy execution, EC2 instance action (stop, terminate, reboot, recover), and Systems Manager OpsCenter incident creation.

### Key alarm configuration parameters:

- Period: The granularity at which the metric is evaluated (60s, 300s, etc.). Higher periods reduce false alarms at the cost of detection latency.

- EvaluationPeriods: The number of periods to evaluate. Setting this to 3 with a 60s period means the alarm fires only if the threshold is violated for 3 consecutive minutes.
- DatapointsToAlarm: Out of EvaluationPeriods periods, how many must be in breach. Setting DatapointsToAlarm=2 with EvaluationPeriods=5 gives 'M of N' evaluation — more resilient to brief spikes.
- TreatMissingData: Controls alarm behavior when metric data is absent. Options: breaching (alarm fires), notBreaching, ignore, and missing. Missing data often indicates the resource is offline — set to breaching for availability alarms.

Composite Alarms: Combine multiple child alarms using Boolean logic (AND, OR, NOT). This is critical for reducing alert noise. Rather than alerting on a single metric spike, a composite alarm can require that CPU utilization is high AND error rate is elevated AND request queue depth is growing before triggering a page. Composite alarms dramatically reduce mean time to detection while avoiding alert fatigue.

## 4.4 Dashboards and Anomaly Detection

CloudWatch Dashboards provide customizable, multi-widget canvases for operational visibility. As of 2025, dashboards support variable substitution, allowing a single dashboard to be parameterized by region, account, or environment — eliminating the need to maintain separate dashboards per environment.

Dashboard widgets can display: metric time series, metric statistics, alarm status, log query results (using embedded Logs Insights queries), custom text (Markdown), and alarm heat maps. The alarm heat map widget is particularly useful for fleet-wide health views — it displays all alarms across a service as a grid of colored cells, enabling instant identification of unhealthy zones.

Anomaly Detection Bands: CloudWatch can model the expected value of any metric using machine learning, then display an anomaly detection band on dashboards and alert when the observed value falls outside the predicted range. The model adapts to weekly seasonality and long-term trends automatically. Anomaly detection alarms are especially useful for business metrics with irregular traffic patterns that are difficult to threshold statically.

## 4.5 Contributor Insights

CloudWatch Contributor Insights provides real-time, automatic analysis of high-cardinality data to identify the top contributors to aggregate metrics. It works by defining rules that specify which log group to analyze and which fields to use as contributor keys.

Common use cases include identifying the top source IPs generating 5xx errors, the highest-volume API callers in CloudTrail data, the Lambda functions with the most cold starts, and the S3 prefixes receiving the most requests. Unlike Logs Insights (which runs on-demand queries), Contributor Insights updates continuously in near-real-time.

## 4.6 Unified Data Store (December 2025)

Note: New in December 2025

Amazon CloudWatch now provides unified management and analytics capabilities for operational, security, and compliance data across your AWS environment and third-party sources. Data can be made available in managed Amazon S3 Tables at no additional storage charge, enabling queries via Athena, Redshift, SageMaker Unified Studio, or any Apache Iceberg-compatible tool.

The unified data store collapses the traditional need for separate data pipelines for each use case. Security analysts can query CloudTrail events and GuardDuty findings alongside VPC flow logs in a single Athena query. DevOps teams can join application logs with deployment events without standing up a custom ETL pipeline. Compliance teams can run cross-service queries to generate evidence for regulators.

AWS sources supported at GA include CloudTrail management and data events, Amazon VPC flow logs, and AWS WAF logs. Third-party managed collectors are available for CrowdStrike, Okta, and Palo Alto Networks. Pipelines can transform and enrich logs to the Open Cybersecurity Schema Framework (OCSF) format for cross-tool compatibility. Organization-wide enablement allows a central security team to configure collection for all accounts in a single operation.

The cost model for the unified data store is important to understand: data stored in S3 Tables is billed at S3 storage rates (significantly cheaper than CloudWatch Logs storage), and Athena queries are billed per TB scanned. For large-scale security analytics workloads, this represents a substantial cost reduction compared to retaining all data in CloudWatch Logs.

## 4.7 Auto-Enablement Rules (April 2026)

Note: New in April 2026

Amazon CloudWatch now supports automatic enablement of Amazon CloudFront Standard access logs, AWS Security Hub CSPM finding logs, and Amazon Bedrock AgentCore memory and gateway logs/traces. Enablement rules can be scoped to the organization, specific accounts, or resource tags — ensuring consistent monitoring coverage without manual setup.

This capability builds on a series of auto-enablement features launched through 2025: VPC flow log auto-enablement (August 2025), organization-wide CloudTrail/VPC/WAF source enablement (December 2025), and the data source targeting for centralization rules (March 2026). Together these features represent AWS's vision of 'zero-touch telemetry': any resource created in a compliant account automatically begins emitting the correct telemetry to the correct destination.

For practical implementation, the recommended approach is: create an enablement rule scoped to your entire organization that automatically enables CloudFront logs, Security Hub CSPM findings, CloudTrail via SLC, and VPC flow logs for all newly created resources matching a production tag. This ensures your telemetry coverage cannot fall behind your infrastructure provisioning rate.

CHAPTER 5

# AWS CloudTrail — API Audit Trail

Every API call, every actor, every timestamp — immutably recorded

## 5.1 Trails vs. Event Data Stores

CloudTrail offers two storage models that serve different use cases and should typically be used together in a mature deployment.

Trails deliver events to an S3 bucket (required) and optionally to CloudWatch Logs and CloudTrail Lake. Trails are the traditional CloudTrail deployment model and are the right choice when you need to integrate with a SIEM that consumes from S3, when you need to retain events in a format compatible with your existing log management infrastructure, or when you need the CloudWatch Logs integration for near-real-time metric filtering and alarms.

Event Data Stores are CloudTrail Lake's native construct — a managed, queryable repository that retains events for up to 7 years (2,557 days) and supports SQL-based analysis via the Lake Query API. Event Data Stores are the right choice for interactive forensic investigation, compliance evidence generation, and workloads that require complex JOINs across event types (e.g., joining CloudTrail events with AWS Config configuration items).

For most organizations, the recommended architecture combines both: a multi-region organization trail delivering to a central logging account S3 bucket for SIEM integration and long-term archival, plus an Event Data Store for interactive investigation with CloudTrail Lake.

Note: Security Requirements

The CloudTrail S3 bucket must have: S3 Object Lock in compliance mode (prevents deletion during retention period), MFA Delete enabled, a bucket policy denying the cloudtrail.amazonaws.com principal from deleting log files, and a KMS customer-managed key for encryption at rest.

## 5.2 Management Events and Data Events

Management events (control-plane events) record operations that modify the configuration of AWS resources: CreateBucket, RunInstances, PutBucketPolicy, CreateUser, and so forth. They are the primary record for compliance auditing, capturing every infrastructure change with the identity of the actor, source IP, user agent, and request/response parameters.

Management events are free for the first copy per region per account. For organization trails, the first copy in the organization is free. Additional copies (e.g., delivering to multiple S3 buckets) incur standard

management event charges.

Data events record resource-level API activity: S3 object reads and writes (GetObject, PutObject, DeleteObject), Lambda function invocations, DynamoDB item-level operations, and others. They are priced per 100,000 events and generate extremely high volume in busy environments — a single S3 bucket serving a large application can generate millions of data events per hour.

Best practice for data events: enable them selectively using advanced event selectors, scoping to specific S3 buckets containing sensitive data (PII, financial records, credentials), specific Lambda functions (payment processors, authentication handlers), and DynamoDB tables containing regulated data. Never enable blanket data event recording for all resources unless the cost and storage implications are fully understood.

## 5.3 Advanced Event Selectors

Advanced Event Selectors (introduced in 2022, continuously enhanced) provide fine-grained control over which events are recorded. They support field-level filtering using the AWS CloudTrail field selector syntax, allowing you to include or exclude events based on account, region, resource ARN, event name, user agent, and more.

```
# CLI: Enable data events only for a specific S3 bucket
aws cloudtrail put-event-selectors \
--trail-name org-compliance-trail \
--advanced-event-selectors '[
{
"Name": "S3-SensitiveBucket-All",
"FieldSelectors": [
{"Field": "eventCategory", "Equals": ["Data"]},
{"Field": "resources.type", "Equals": ["AWS::S3::Object"]},
{"Field": "resources.ARN", "StartsWith": [
"arn:aws:s3:::pci-cardholder-data/"
]}
]
},
{
"Name": "All-Management-Events",
"FieldSelectors": [
{"Field": "eventCategory", "Equals": ["Management"]}
]
}
]'
```

Code 5.1 — CLI: advanced event selectors to capture selective data events.

## 5.4 CloudTrail Insights — Now Covering Data Events (November 2025)

Note: New in November 2025

AWS extended CloudTrail Insights to data events. Insights now establishes normal baselines for both management and data access patterns, automatically detecting anomalies such as unexpected surges in S3 DeleteObject API calls or increased Lambda invocation error rates — without requiring you to build custom detection logic.

CloudTrail Insights works by establishing a rolling baseline of normal API call rates and error rates for your account, then publishing an Insights event when it detects a statistically significant deviation from that baseline. When an anomaly is detected, CloudTrail provides the raw data events from the anomaly period, enabling rapid forensic investigation.

Insights events appear in the CloudTrail console under the Insights tab, are delivered to the same S3 bucket as your trail, and can trigger EventBridge rules for automated response. The extension to data events is particularly valuable for detecting S3 exfiltration patterns (sudden surge in GetObject calls from new principals or regions) and Lambda abuse (unusual spike in invocation count from anomalous callers).

## 5.5 CloudTrail Lake — SQL Queries

CloudTrail Lake transforms CloudTrail from an audit log store into a queryable security data lake. Event Data Stores can be queried using standard SQL SELECT statements. Queries can span multiple Event Data Stores (e.g., joining CloudTrail events with AWS Config configuration items stored in a separate Event Data Store) and can aggregate across time periods up to the retention horizon.

```
-- Security investigation: All actions by a specific IAM principal in 7 days
SELECT eventTime, eventSource, eventName, sourceIPAddress,
requestParameters, responseElements, errorCode, errorMessage
FROM eds-arn-here
WHERE userIdentity.arn = 'arn:aws:iam::123456789012:user/suspicious-user'
AND eventTime > DATE_ADD('day', -7, NOW())
ORDER BY eventTime DESC;

-- Compliance: Find all S3 bucket policy changes in the last 90 days
SELECT eventTime, recipientAccountId,
userIdentity.principalId AS actor,
requestParameters.bucketName AS bucket,
sourceIPAddress
FROM eds-arn-here
WHERE eventName IN ('PutBucketPolicy', 'DeleteBucketPolicy', 'PutBucketAcl')
```

```
AND eventTime > DATE_ADD('day', -90, NOW())
ORDER BY eventTime;

-- Threat hunting: Unusual AssumeRole chains (role-hopping)
SELECT eventTime, userIdentity.sessionContext.sessionIssuer.arn AS parent_role,
requestParameters.roleArn AS assumed_role, sourceIPAddress
FROM eds-arn-here
WHERE eventName = 'AssumeRole'
AND userIdentity.type = 'AssumedRole'
AND eventTime > DATE_ADD('hour', -24, NOW());
```

Code 5.2 — CloudTrail Lake SQL: security investigation and compliance queries.

## 5.6 Integrity Validation and Security

CloudTrail log file validation creates a SHA-256 digest of each log file and signs a chain of digests using an RSA-2048 key held by AWS. The digest files are stored separately from the log files. This chain-of-custody mechanism allows you to prove that logs have not been tampered with after delivery to S3 — critical for regulatory compliance and incident investigations where the integrity of the audit trail may be challenged.

```
# Validate CloudTrail log integrity for a time range
aws cloudtrail validate-logs \
--trail-arn arn:aws:cloudtrail:us-east-1:123456789012:trail/org-compliance-trail \
--start-time 2026-01-01T00:00:00Z \
--end-time 2026-04-01T00:00:00Z \
--verbose

# Expected output for unmodified logs:
# Logs validated: 2,847
# Logs with validation errors: 0
```

Code 5.3 — CLI: CloudTrail log integrity validation.

Additional security controls for CloudTrail: enable S3 Object Lock in compliance mode on the log bucket; configure an S3 bucket policy that explicitly denies s3:DeleteObject and s3:DeleteBucket for the log bucket; enable S3 MFA Delete; encrypt logs with a KMS CMK with a restrictive key policy; and set up CloudWatch alarms on StopLogging, DeleteTrail, and UpdateTrail API calls via CloudTrail Insights or metric filters.

## 5.7 CloudTrail + CloudWatch Integration (December 2025)

Note: New in December 2025

AWS launched simplified enablement of CloudTrail events in CloudWatch via service-linked channels (SLCs). This allows CloudTrail events to flow into CloudWatch Logs alongside other log

sources without requiring a trail configuration. SLCs provide termination protection — CloudWatch prevents disabling collection if doing so would violate retention policies.

The SLC-based integration enables CloudWatch metric filters to fire on specific API calls within seconds of their occurrence — significantly faster than the traditional trail-to-S3-to-Lambda pipeline. Combined with CloudWatch alarms, this enables near-real-time detection of critical API calls like IAM policy changes, Security Group modifications, or CloudTrail disabling attempts.

For CIS AWS Foundations Benchmark compliance, recommendations 4.1–4.15 require CloudWatch metric filters and alarms for specific CloudTrail event patterns (unauthorized API calls, root account usage, IAM policy changes, CloudTrail configuration changes, S3 bucket policy changes, and others). The SLC integration makes this much simpler to configure and maintain.

CHAPTER 6

# AWS Config — Configuration Compliance

Continuous visibility into what changed, who changed it, and whether it complies

## 6.1 Configuration Recorder and Delivery Channel

AWS Config records the state of supported AWS resources at the time of every change (and periodically for resources that do not generate change events). The Configuration Recorder determines which resource types are tracked. The Delivery Channel specifies the S3 bucket for configuration snapshots and configuration history, and the SNS topic for change notifications.

When a resource is created, modified, or deleted, AWS Config generates a configuration item (CI) containing: the complete resource configuration in JSON format, all resource relationships (which VPC this EC2 instance is in, which subnets the ELB spans), the AWS account and region, the time of change, and the user identity from CloudTrail that made the change.

The CI history for any resource provides a complete timeline of its configuration evolution — invaluable for root cause analysis ('what changed about this security group in the 24 hours before the breach?') and compliance auditing ('was encryption enabled on this S3 bucket throughout the audit period?').

In AWS Organizations, enabling Config with delegated administration means the management account designates a dedicated security or audit account. That account can enable Config in all member accounts, define organization-wide Config rules, deploy Conformance Packs, and view aggregate compliance data across the fleet.

## 6.2 Managed and Custom Config Rules

Config Rules evaluate resource configurations against a desired-state definition and report COMPLIANT or NON_COMPLIANT for each evaluated resource. AWS provides over 300 managed rules (predefined, continuously updated). Custom rules are Lambda functions that receive a configuration item and return compliance verdicts — used for organization-specific policies.

| Rule Identifier | What It Checks | Relevant Framework |
|---|---|---|
| CLOUD_TRAIL_ENABLED | CloudTrail is active in all regions | SOC 2, PCI DSS, HIPAA |
| CLOUD_TRAIL_LOG_FILE_VALIDATION_ENABLED | Log file validation is on | PCI DSS Req 10.5 |
| CLOUD_TRAIL_ENCRYPTION_ENABLED | CloudTrail logs encrypted with KMS CMK | PCI DSS, HIPAA |

| Rule Identifier | What It Checks | Relevant Framework |
|---|---|---|
| GUARDDUTY_ENABLED_CENTRALIZED | GuardDuty enabled via Organizations | SOC 2, CIS 2.1 |
| SECURITYHUB_ENABLED | Security Hub is enabled | CIS 2.2 |
| VPC_FLOW_LOGS_ENABLED | Flow logs exist for all VPCs | HIPAA, PCI DSS |
| MFA_ENABLED_FOR_IAM_CONSOLE_ACCESS | IAM console users have MFA | CIS 1.10, SOC 2 |
| IAM_ROOT_ACCESS_KEY_CHECK | Root account has no active access keys | CIS 1.4 |
| S3_BUCKET_PUBLIC_READ_PROHIBITED | No S3 bucket allows public read | GDPR, HIPAA, PCI |
| EC2_EBS_ENCRYPTION_BY_DEFAULT | EBS default encryption enabled | PCI DSS, HIPAA |
| CLOUDWATCH_LOG_GROUP_ENCRYPTED | Log groups encrypted with KMS | HIPAA, SOC 2 |
| MULTI_REGION_CLOUD_TRAIL_ENABLED | Multi-region trail exists and is logging | CIS 3.1, PCI DSS |
| ROOT_ACCOUNT_MFA_ENABLED | Root account has MFA configured | CIS 1.5, SOC 2 |
| IAM_PASSWORD_POLICY | Password policy meets complexity requirements | SOC 2, PCI DSS |

Table 6.1 — Critical AWS Config managed rules by compliance framework.

## 6.3 Conformance Packs

A Conformance Pack is a collection of Config rules and optional remediation actions packaged as a CloudFormation YAML template, deployable at scale across an organization via a single API call. AWS provides sample Conformance Packs pre-mapped to CIS AWS Foundations Benchmark v1.4, PCI DSS, HIPAA, NIST 800-53 Revision 5, and the AWS Foundational Security Best Practices standard.

Organizations can customize sample Conformance Packs or build their own. A custom pack might encode all of your company's internal configuration standards (e.g., 'all production EC2 instances must be in a VPC, have detailed monitoring enabled, and not have public IP addresses') as a deployable, version-controlled template.

Conformance Pack compliance reports show the aggregate COMPLIANT/NON_COMPLIANT count for each rule, the list of non-compliant resources, and the compliance trend over time. These reports can be exported and submitted directly to auditors as evidence of configuration compliance.

## 6.4 Remediation Actions

Config Rules can be paired with Remediation Actions that automatically correct non-compliant resources using Systems Manager Automation documents. Remediation can be manual (operator-initiated from the Config console) or automatic (triggered immediately when a resource becomes non-compliant).

> Warning: Warning: Automatic Remediation
>
> Automatic remediation is powerful but dangerous if misconfigured. Always test remediation documents in a non-production account first. Incorrect automatic remediation can modify or delete production resources. Consider using notification-only remediation (Lambda posts to Slack) until confidence in the remediation logic is high.

### Commonly used AWS-provided SSM Automation documents for remediation:

- AWS-EnableCloudTrail: Creates a trail if CloudTrail is not enabled.
- AWS-EnableVpcFlowLogs: Creates flow logs for non-compliant VPCs.
- AWS-RevokeUnusedIAMUserCredentials: Deactivates unused IAM credentials older than a configurable threshold.
- AWS-EnableS3BucketEncryption: Enables SSE-S3 or SSE-KMS on a non-encrypted bucket.
- AWS-SetS3BucketPublicAccessBlock: Blocks all public access on a publicly accessible bucket.
- AWS-EnableGuardDuty: Enables GuardDuty in accounts where it is not enabled.

## 6.5 Multi-Account Config Aggregator

An AWS Config Aggregator collects configuration data and compliance status from multiple accounts and regions into a centralized account. With Organizations delegated administration, the security account sees a fleet-wide view: how many resources are non-compliant with each rule, across all accounts and regions, in a single dashboard.

The aggregator enables fleet-wide queries using the AWS Config Advanced Query feature: SQL SELECT statements against the aggregated configuration database. You can query 'show me all EC2 instances across all accounts that are not in a VPC' or 'find all S3 buckets with public access not blocked, in any account, in any region' — instantly.

## 6.6 AWS Audit Manager

> Warning: Important Notice
>
> AWS Audit Manager will no longer be open to new customers starting April 30, 2026. Organizations already using Audit Manager retain access. New customers should evaluate AWS Security Hub's compliance features and automated compliance platforms such as Drata, Vanta, or Secureframe.

For existing users: Audit Manager continuously collects evidence from Config evaluations, CloudTrail events, Security Hub findings, and AWS API calls, mapping each piece of evidence to specific controls in your chosen compliance framework (SOC 2, PCI DSS, HIPAA, or NIST 800-53). Assessments produce evidence reports ready for external auditors.

The transition away from Audit Manager is a significant shift. Organizations should begin migrating to alternative evidence collection pipelines now: either native Security Hub compliance standards (which replaced much of Audit Manager's CSPM functionality) or third-party automated compliance platforms that integrate with AWS natively via API.

CHAPTER 7

# Threat Detection and Security Monitoring

GuardDuty, Security Hub, Inspector, Macie, and the new Security Agent

## 7.1 Amazon GuardDuty

Amazon GuardDuty is a managed threat detection service that uses machine learning, anomaly detection, and threat intelligence feeds to identify malicious or unauthorized behavior in your AWS environment. It operates as a passive observer — analyzing log telemetry that already flows through AWS infrastructure — without requiring agents, network probes, or changes to your applications.

GuardDuty findings are classified by threat category and include a confidence score (1–10), affected resource details, actor context (IP address, ASN, user agent), and evidence from the underlying telemetry. Findings are published to the GuardDuty console, S3 (for archival), EventBridge (for automation), and Security Hub (for aggregation).

The base GuardDuty service analyzes CloudTrail management events, VPC flow logs, and DNS logs. Protection plans extend coverage:

| Protection Plan | Telemetry Analyzed | Key Finding Types |
|---|---|---|
| S3 Protection | S3 data event logs | Data:S3/MaliciousIPCaller, Policy:S3/BucketPublicAccessGranted |
| EKS Audit Log | EKS control plane audit logs | Policy:Kubernetes/AdminAccessToDefaultServiceAccount |
| EKS Runtime | eBPF agent on Kubernetes nodes | Execution:Runtime/MaliciousFileExecution, Backdoor:Runtime/C2Activity |
| Malware Protection | EBS snapshot analysis on EC2/ECS | Execution:EC2/MaliciousFileExecuted |
| RDS Protection | RDS login activity | CredentialAccess:RDS/AnomalousBehavior.SuccessfulLogin |
| Lambda Protection | Lambda network activity | Backdoor:Lambda/C2Activity, CryptoCurrency:Lambda/BitcoinTool.B |
| Runtime Monitoring | eBPF-based process and file events | Execution:Runtime/ReverseShell, Impact:Runtime/CryptoMinerExecuted |

Table 7.1 — GuardDuty protection plans and key finding types.

## 7.2 Extended Threat Detection (re:Invent 2025)

Note: New at re:Invent 2025

GuardDuty Extended Threat Detection expanded with new attack sequence findings for EC2 instances and ECS tasks, building on existing coverage for IAM, S3, and EKS. The system uses AI/ML trained at cloud scale to link related security signals into high-confidence multistage attack findings.

Extended Threat Detection represents a paradigm shift from isolated findings (this IP is suspicious; this API call is unusual) to correlated attack sequences (this identity was compromised, then used to enumerate the environment, then exfiltrated data from S3 over the following 4 hours). These attack sequence findings map directly to MITRE ATT&CK; tactics and techniques.

A typical extended threat detection finding timeline might include: an InitialAccess finding from an unusual CloudTrail login, followed by a Discovery finding from rapid DescribeInstances calls, followed by a LateralMovement finding from RunInstances in an unusual region, followed by an Exfiltration finding from large S3 GetObject calls. Extended Threat Detection presents all of these as a single coherent attack story, reducing the analyst's cognitive load and time to containment.

## 7.3 AWS Security Hub (GA — re:Invent 2025)

Note: Generally Available — December 2025

The redesigned AWS Security Hub reached general availability at re:Invent 2025. It delivers near real-time risk analytics with up to one year of historical trend data, period-over-period analysis, attack path visualization, and cross-region aggregation. Now available in GovCloud as of March 2026.

Security Hub v2 organizes security signals into four operational categories: Threats (active indicators from GuardDuty), Exposures (vulnerabilities from Inspector), Resources (misconfigurations from CSPM), and Coverage (gaps in security service deployment across accounts/regions). This structure maps directly to incident response and remediation workflows.

Attack Path Visualization: Security Hub automatically constructs attack path graphs showing how an adversary could chain threats, vulnerabilities, and misconfigurations to compromise a critical resource. The graph is continuously updated as new findings arrive, and includes probability scoring based on the accessibility and exploitability of each node.

Automated Response Workflows: Custom actions in Security Hub allow security analysts to trigger Lambda functions directly from the Security Hub console when reviewing findings — without switching to EventBridge or the Lambda console. Pre-built actions include 'Isolate Instance', 'Revoke Credentials', 'Create Incident', and 'Send to SOAR'.

ASFF Normalization: All findings in Security Hub are normalized to the AWS Security Finding Format. This normalization enables cross-source correlation — a high-severity GuardDuty finding can be automatically correlated with an Inspector vulnerability finding on the same EC2 instance, producing a combined risk score that accounts for both active exploitation and exploitability.

## 7.4 Amazon Inspector (March 2026 Updates)

Note: New in March 2026

Amazon Inspector expanded agentless EC2 scanning with Windows OS vulnerability support and KB-based findings. Enhanced detection covers WordPress, Apache HTTP Server, Python packages, and Ruby gems — all without requiring an agent installation.

Inspector's scanning architecture supports two modes for EC2: agent-based scanning (using SSM Agent for real-time software inventory) and agentless scanning (using EBS snapshot analysis without any software installation). The hybrid default uses agent-based for SSM-managed instances and automatically falls back to agentless for unmanaged instances — ensuring coverage regardless of instance configuration.

Windows KB-based findings: Rather than generating a separate finding for each CVE addressed by a single Microsoft patch, Inspector now groups related CVEs under a single consolidated KB finding. Each KB finding shows the highest CVSS score and EPSS score from its constituent CVEs, plus a direct link to the Microsoft KB article. This dramatically reduces finding noise for Windows administrators.

Code Security (GA since June 2025): Inspector can scan application source code directly, before build. This integrates with CI/CD pipelines (CodePipeline, Jenkins, GitHub Actions) to catch vulnerabilities at the earliest possible stage. Code Security findings use the same ASFF format and integrate with Security Hub, enabling a unified view of code, package, and runtime vulnerabilities.

SBOM Export: Inspector generates a Software Bill of Materials (SBOM) for each scanned resource, listing all detected software packages and their versions. SBOMs can be exported in SPDX or CycloneDX format and are increasingly required by enterprise procurement and government contracts.

## 7.5 Amazon Macie

Amazon Macie is a data security service that uses machine learning to discover, classify, and protect sensitive data in Amazon S3. Macie maintains a continuously updated inventory of all S3 buckets in your account, assessing each for public exposure, cross-account access, server-side encryption status, and replication configuration.

Sensitive data discovery jobs run on a scheduled or on-demand basis, scanning the content of S3 objects for over 100 managed data identifiers covering: PII (Social Security numbers, passport numbers, driver's license numbers, email addresses, phone numbers), financial data (credit card numbers, bank routing numbers, IBAN), healthcare data (medical record numbers, DEA registrant numbers), and credentials (private keys, API keys, OAuth tokens).

Custom data identifiers allow organizations to define additional patterns using regular expressions — for example, matching internal employee IDs, customer reference numbers, or proprietary document markers. Findings from custom identifiers appear in the same workflow as managed identifier findings.

For GDPR compliance, Macie is particularly valuable for data subject access requests (DSARs) — you can run a targeted Macie job on a specific S3 prefix to locate all objects containing a particular customer's identifying information.

## 7.6 AWS Security Agent (Preview — December 2025)

Note: Preview — December 2025

AWS Security Agent is a new AI-powered service that performs automated application security reviews and adaptive penetration testing. It dynamically builds attack plans based on application design documents, source code, and internal security policies, and adjusts its testing as it discovers new endpoints and credentials.

Security Agent fills the gap between static analysis (Inspector Code Security) and runtime detection (GuardDuty). It performs active behavioral testing — logic flaws, authentication bypasses, authorization escalations, and business logic vulnerabilities that cannot be found by examining code or packages alone.

At preview stage, Security Agent supports web application security testing for applications hosted on AWS. The service requires a deployment-time permission to read application design documents from S3 and access source code from CodeCommit or CodeBuild. Testing is performed in an isolated environment using production-equivalent credentials.

## 7.7 VPC Flow Logs and AWS WAF Logging

VPC Flow Logs capture network interface metadata at the VPC, subnet, or ENI level. Version 3+ flow logs include additional fields: vpc-id, subnet-id, instance-id, tcp-flags, type (IPv4/IPv6/EFA), pkt-srcaddr, and pkt-dstaddr. These additional fields significantly improve network forensics capability — you can determine the actual source and destination addresses even for traffic passing through NAT gateways.

As of August 2025, CloudWatch can automatically enable VPC flow logs for all VPCs across an organization using telemetry enablement rules scoped by account, OU, or resource tag. This eliminates the most common flow log coverage gap (VPCs created in new accounts that are not yet covered by flow logging).

AWS WAF logging captures the complete request body, all matched rules, and the final allow/block decision for every web request processed by a Web ACL. As of September 2025, AWS WAF includes 500 MB of CloudWatch Logs ingestion per million WAF requests at no charge — making it cost-effective to enable full WAF logging for most workloads. Data protection controls (February 2025) allow sensitive request fields (authorization headers, cookies, query string parameters) to be cryptographically hashed or redacted before log storage, addressing GDPR and PCI data minimization requirements.

CHAPTER 8

# Distributed Tracing and Application Observability

Following every request from the edge to the database and back

## 8.1 AWS X-Ray — Architecture and Sampling

AWS X-Ray provides end-to-end distributed tracing for applications built on AWS. A trace represents the entire path of a request: from an incoming API Gateway call, through Lambda function invocations and internal service calls, to DynamoDB reads, and back. Each hop in the request path is a segment; sub-operations within a segment (a DynamoDB call within a Lambda function) are subsegments. Together, the hierarchy of segments and subsegments provides nanosecond-resolution visibility into every step of a request's execution.

The X-Ray Service Map provides a real-time graph visualization of all services and their dependencies, with latency distribution and error rate overlaid on each node and edge. This map updates in near-real-time and is invaluable for identifying performance bottlenecks and error propagation patterns in microservices architectures.

Sampling is critical for production cost management. By default, X-Ray records the first request per second and 5% of subsequent requests. Custom sampling rules are matched in priority order and can specify different rates for different URL paths, HTTP methods, hosts, and service names. For example: record 100% of requests to /checkout (business-critical, low volume) and 1% of requests to /health (high frequency, low diagnostic value).

```
# X-Ray / ADOT sampling rule: high fidelity for checkout, low for health
{
"version": 2,
"rules": [
{
"description": "Checkout - 100% sampling",
"host": "*",
"http_method": "POST",
"url_path": "/checkout*",
"fixed_target": 10,
"rate": 1.0
},
{
"description": "Health checks - 0% sampling",
```

```
"host": "*",
"http_method": "GET",
"url_path": "/health",
"fixed_target": 0,
"rate": 0.0
}
],
"default": {
"fixed_target": 1,
"rate": 0.05
}
}
```

Code 8.1 — X-Ray sampling rules: 100% checkout, 0% health checks, 5% default.

## 8.2 Transition to OpenTelemetry (February 2026)

Note: Major Transition — February 2026

The AWS X-Ray SDKs and Daemon entered maintenance mode on February 25, 2026. They will receive only security updates until end-of-support in February 2027. AWS now recommends ADOT and OpenTelemetry SDKs as the primary instrumentation path for all new development.

The key architectural change: previously, developers used X-Ray SDKs to instrument their application code and the X-Ray Daemon as a local agent to buffer and forward trace segments to the X-Ray service. Now, developers use OpenTelemetry SDKs for instrumentation and the ADOT Collector (either as a Lambda layer, ECS sidecar, EKS add-on, or standalone process) to export traces to X-Ray.

The X-Ray console, Service Map, trace storage, and all query capabilities remain fully supported and continue to receive new features. The transition is client-side only — traces from both X-Ray SDKs and ADOT are stored and visualized identically in the X-Ray backend.

Benefits of the transition: OpenTelemetry's semantic conventions ensure consistent attribute naming across all instrumented systems (database call spans always use db.system, db.name, db.statement); auto-instrumentation libraries are maintained by a much larger community; traces can be simultaneously exported to X-Ray, Datadog, Honeycomb, Jaeger, or any OTLP endpoint without code changes — only the Collector configuration changes.

## 8.3 AWS Distro for OpenTelemetry (ADOT)

ADOT is AWS's production-supported, performance-tested distribution of the OpenTelemetry Collector and SDKs. It includes AWS-specific components not yet available in the upstream OpenTelemetry distribution: the AWS X-Ray Exporter, the AWS EMF Exporter (for CloudWatch metrics), the AWS Container Insights Receiver, and AWS-specific resource detectors.

```
# ADOT Collector config: traces to X-Ray, metrics to CloudWatch EMF
receivers:
otlp:
protocols:
grpc:
endpoint: 0.0.0.0:4317
http:
endpoint: 0.0.0.0:4318

processors:
batch:
timeout: 1s
send_batch_size: 1024
resourcedetection:
detectors: [env, ecs, ec2]
timeout: 5s

exporters:
awsxray:
region: us-east-1
local_mode: false # use IMDS, no daemon required
awsemf:
region: us-east-1
log_group_name: /otel/metrics
namespace: MyApp/Production
dimension_rollup_option: NoDimensionRollup

service:
pipelines:
traces:
receivers: [otlp]
processors: [resourcedetection, batch]
exporters: [awsxray]
metrics:
receivers: [otlp]
processors: [resourcedetection, batch]
exporters: [awsemf]
```

Code 8.2 — ADOT Collector: dual export to X-Ray (traces) and CloudWatch EMF (metrics).

## 8.4 CloudWatch Transaction Search

CloudWatch Transaction Search is a newer capability that correlates logs, metrics, and traces by transaction ID across log groups and services — without requiring full ADOT instrumentation. It surfaces all log events sharing a common transaction context, enabling drill-down from a CloudWatch alarm directly into the specific log lines from every service involved in the affected transactions.

Transaction Search is particularly valuable in heterogeneous environments where not all services are fully instrumented with ADOT. As long as a transaction ID (correlation ID, request ID, or X-Ray trace ID) is present in log events, Transaction Search can correlate them across services and accounts.

## 8.5 Container Observability: ECS and EKS

ECS Container Insights: Provides CPU, memory, network, and storage metrics for clusters, services, tasks, and containers. Enable at the cluster level using the --settings flag or by enabling containerInsights for the cluster resource. Container Insights delivers metrics to CloudWatch using the EMF format — each metric is also available as a log event in the /aws/ecs/containerinsights log group, enabling detailed drill-down with Logs Insights.

For traces in ECS, deploy the ADOT Collector as a sidecar container in your task definition. All application containers in the task share the task's local network, so the ADOT sidecar is reachable at localhost:4317 (gRPC) or localhost:4318 (HTTP). The ADOT ECS container image is maintained by AWS and available in Amazon ECR Public.

EKS Container Insights: The Amazon CloudWatch Observability add-on (installed via EKS add-on management or Helm) deploys CloudWatch Agent and Fluent Bit as DaemonSets in your cluster. CloudWatch Agent collects cluster-level metrics (CPU, memory, network, disk I/O) for nodes, pods, and containers. Fluent Bit collects pod logs and routes them to CloudWatch Logs. The add-on also enables X-Ray tracing support for pods by default.

EKS Control Plane logs (API Server, Audit, Authenticator, Controller Manager, Scheduler) can be enabled per cluster and are delivered to CloudWatch Logs in the /aws/eks/cluster-name/cluster log group. The audit log in particular is critical for security monitoring — it records every Kubernetes API call with the requesting user's identity.

## 8.6 Serverless Observability: Lambda

Lambda is simultaneously the most instrumented and the hardest to observe function compute environment on AWS. The runtime automatically emits START, END, and REPORT log lines to CloudWatch Logs for every invocation. The REPORT line includes: duration (ms), billed duration (ms), memory size (MB), max memory used (MB), and — for cold starts — init duration (ms). This baseline telemetry is sufficient for cost optimization and basic performance monitoring.

For production observability, the recommended Lambda stack includes:

- EMF-formatted custom metrics for business-level KPIs (OrdersProcessed, PaymentsFailed) with dimensions for function version and environment.
- Structured JSON logging to CloudWatch Logs with a correlation/trace ID in every log event.
- ADOT Lambda Layer for automatic instrumentation of AWS SDK calls (DynamoDB, S3, SQS) and outbound HTTP requests.
- Lambda Powertools for Python/TypeScript/Java: a community-maintained library providing structured logging, metrics (via EMF), tracing (via ADOT), and feature flags in a single package.
- CloudWatch Lambda Insights: Enhanced monitoring extension that collects system-level metrics (CPU, memory, network, disk I/O) and publishes them to CloudWatch at each invocation. Particularly valuable for identifying memory leaks and CPU-bound functions.

CHAPTER 9

# Multi-Account and Multi-Region Monitoring

Centralizing telemetry across an AWS Organization

## 9.1 AWS Organizations and Control Tower

Large enterprises operate dozens to hundreds of AWS accounts — dedicated accounts for production, staging, development, security tooling, shared networking, DNS, audit logging, and individual business units or product teams. This account-per-workload model is the AWS recommended pattern: it provides the strongest isolation boundary, the cleanest cost attribution, and the simplest permission model.

AWS Organizations provides the governance framework: policy-based management using Service Control Policies (SCPs), centralized billing, and the delegation model that underpins all multi-account security services. AWS Control Tower provides an automated landing zone with guardrails, an account vending machine, and a pre-configured account baseline.

For observability, the key Organizations-level configurations are:

- Designate a dedicated Log Archive account to receive all CloudTrail organization trails and Config delivery channels. This account should have extremely restricted access — only the audit team should be able to read from it, and nobody should be able to modify or delete logs.
- Designate a dedicated Security Tooling account as the delegated administrator for GuardDuty, Security Hub, Inspector, Macie, Config, and CloudTrail. This account becomes the single pane of glass for security findings across the organization.
- Use SCPs to prevent member accounts from disabling CloudTrail, disabling GuardDuty, or modifying the organization trail — even by account administrators.

## 9.2 CloudWatch Observability Access Manager (OAM)

CloudWatch OAM enables cross-account and cross-region observability without copying or moving data. Source accounts create OAM Links to a monitoring account's Sink. The monitoring account can then query metrics, logs, and X-Ray traces from all linked accounts directly — as if they were in the same account.

```
# In monitoring account: create the OAM Sink
aws oam create-sink --name "org-monitoring-sink" --region us-east-1

# Allow entire org to link (replace with your org ID)
aws oam put-sink-policy \
--sink-identifier "arn:aws:oam:us-east-1:999888777666:sink/abc123" \
```

```
--policy '{
"Version": "2012-10-17",
"Statement": [{
"Effect": "Allow", "Principal": "*",
"Action": ["oam:CreateLink", "oam:UpdateLink"], "Resource": "*",
"Condition": {"ForAnyValue:StringEquals": {
"aws:PrincipalOrgID": "o-exampleorg123"}}
}]}'

# In each source account: create a Link to the Sink
aws oam create-link \
--label-template '$AccountName' \
--resource-types '[
"AWS::CloudWatch::Metric",
"AWS::Logs::LogGroup",
"AWS::XRay::Trace",
"AWS::ApplicationInsights::Application"
]' \
--sink-identifier "arn:aws:oam:us-east-1:999888777666:sink/abc123"

# Deploy to all org accounts using CloudFormation StackSets
aws cloudformation create-stack-set \
--stack-set-name oam-monitoring-link \
--template-body file://oam-link.yaml \
--permission-model SERVICE_MANAGED \
--auto-deployment Enabled=true,RetainStacksOnAccountRemoval=false
```

Code 9.1 — CloudWatch OAM: cross-account observability setup.

## 9.3 Centralized CloudTrail and Config

Organization CloudTrail: Create a single trail in the management account with IsOrganizationTrail=true and IsMultiRegionTrail=true. This trail automatically captures events from all existing and future member accounts in all regions, delivering to a central S3 bucket in the Log Archive account. Member accounts cannot disable or modify an organization trail — they can create their own additional trails but cannot affect the organization trail.

Organization Config Aggregator: An AWS Config Aggregator in the Security Tooling account (as delegated administrator) collects configuration items and rule compliance data from all member accounts and regions. The Advanced Query feature enables fleet-wide SQL queries against the aggregated configuration database.

Config Recorder in All Accounts: Each member account must have a Config Recorder enabled to generate the configuration items that the Aggregator collects. Use AWS Config's organization-wide

enablement to automatically enable the recorder in all accounts. Organizations Control Tower landing zones include Config enablement by default.

## 9.4 Security Hub Aggregation

Security Hub cross-region aggregation collects all findings from all enabled regions into a single aggregation region. This should be configured on day one — findings generated before aggregation is configured are not retroactively replicated.

With Organizations delegated administration, the Security Tooling account becomes the hub for all member account findings. The delegated admin can: enable Security Hub in all member accounts, configure standards (CIS, PCI, NIST) across all accounts, set organization-wide suppression rules for known-false-positive finding types, and build cross-account dashboards showing overall security posture.

The Security Hub Summary Dashboard (new in the December 2025 GA release) provides up to one year of historical trend data for cross-account security posture, period-over-period comparison (how has our critical finding count changed vs. last month?), and severity breakdown across all accounts in the organization.

## 9.5 Log Centralization Patterns (2026)

As of Q1 2026, AWS provides three complementary patterns for centralizing logs, each suited to different use cases and query needs:

- Pattern 1 — CloudWatch Telemetry Config Rules: Organization-wide enablement rules automatically configure sources (VPC flow logs, CloudFront access logs, Security Hub CSPM findings, Bedrock logs) and route them to a designated CloudWatch Logs destination. Data source targeting (March 2026) allows rules to be scoped to specific source types. Best for: operational observability where CloudWatch Insights queries are the primary access pattern.
- Pattern 2 — S3 Aggregation with Iceberg: CloudTrail organization trails and Config delivery channels deliver to a central Log Archive S3 bucket. Using CloudWatch's December 2025 unified data store, these logs are materialized into Iceberg tables queryable via Athena. Best for: long-term retention, compliance evidence, and cross-joining CloudTrail with Config for audit queries.
- Pattern 3 — Kinesis Firehose to SIEM: CloudWatch Log subscriptions via Kinesis Firehose stream normalized telemetry to Splunk, Microsoft Sentinel, Elastic SIEM, or Amazon OpenSearch Service in near-real-time. Best for: organizations with existing SIEM investments that need AWS telemetry correlated with on-premises data.

Most mature organizations use all three patterns simultaneously: Pattern 1 for operational monitoring, Pattern 2 for compliance evidence, and Pattern 3 for security operations center tooling.

## 9.6 Network Monitoring at Scale

CloudWatch Network Monitoring (Network Flow Monitors) provides near-real-time network performance metrics for workloads spanning EC2, EKS, RDS, S3, and DynamoDB. As of May 2025, flow monitors support multi-account monitoring via Organizations integration — a single network administrator in the monitoring account can enable visibility into network paths crossing multiple accounts.

For Direct Connect connectivity, AWS Direct Connect added CloudWatch BGP monitoring metrics in March 2026: VirtualInterfaceBgpStatus (detects session failures), VirtualInterfaceBgpPrefixesAccepted (tracks prefix count from on-premises), and VirtualInterfaceBgpPrefixesAdvertised (shows prefixes advertised to AWS). These metrics enable proactive alerting before BGP session failures cause connectivity outages.

Route 53 Resolver query logging, when combined with VPC flow logs and DNS Firewall, provides complete north-south DNS visibility: you can see every DNS query from every instance in your VPCs, the response it received, and whether DNS Firewall blocked or allowed it — critical for detecting DNS-based exfiltration and C2 communication.

CHAPTER 10

# Compliance, Best Practices & Operational Maturity

From passing an audit to maintaining a continuously compliant cloud

## 10.1 Regulatory Frameworks

AWS operates under over 140 security standards and compliance certifications, including SOC 1/2/3, ISO 27001, PCI DSS Level 1, FedRAMP Moderate/High, and HIPAA. AWS publishes its compliance reports on AWS Artifact (free, self-service). However, holding the underlying infrastructure certifications does not confer compliance on your workloads — you inherit the controls that operate at the infrastructure level, but you are responsible for the controls that operate at the application and data level.

| Framework | Core Monitoring/Audit Requirements | Primary AWS Controls |
|---|---|---|
| SOC 2 Type II | Availability monitoring; access log retention 90d+; change management; anomaly detection | CloudWatch, CloudTrail, Config, GuardDuty, Security Hub |
| PCI DSS v4.0 | Log all CHD access; retain 12 months; SIEM alerting; FIM; daily log review | CloudTrail (data events), Flow Logs, Config, GuardDuty, Inspector |
| HIPAA Security Rule | Audit controls; automatic logoff; PHI access tracking; 6-year retention | CloudTrail, CloudWatch Logs, Macie, KMS, S3 Object Lock |
| GDPR Art. 25/32 | Breach notification 72h; data minimization; access controls; data subject rights | Macie, CloudTrail, GuardDuty, Config, KMS |
| FedRAMP High | NIST 800-53 Rev 5 High baseline; ConMon; vulnerability scanning | Security Hub (NIST pack), Config, Inspector, CloudTrail |
| ISO 27001 | A.12.4: logging; A.12.7: audit; A.16: incident management | CloudTrail, Config, Security Hub, GuardDuty |

Table 10.1 — Major compliance frameworks and AWS monitoring controls.

### PCI DSS v4.0 — Special Considerations

PCI DSS v4.0 became the only acceptable standard on April 1, 2025. Key changes from v3.2.1 relevant to AWS monitoring include: Requirement 10.3 now mandates automated mechanisms that detect and protect logs from unauthorized modifications; Requirement 10.7 requires formal procedures for detecting and responding to log failures; Requirement 12.10.7 requires detecting and responding to payment page skimming attacks (use WAF logging + Macie + CloudTrail data events).

## 10.2 CIS AWS Foundations Benchmark

The CIS AWS Foundations Benchmark v3.0 (current as of 2025) provides 58 recommendations across identity, storage, logging, monitoring, and networking. Security Hub includes a built-in CIS standard that evaluates your environment against all recommendations in real time. The benchmark is divided into Level 1 (essential, low operational impact) and Level 2 (defense-in-depth, may require additional configuration).

### Level 1 monitoring and logging requirements (critical):

- 3.1: Ensure CloudTrail is enabled in all regions (Config: MULTI_REGION_CLOUD_TRAIL_ENABLED).
- 3.2: Ensure CloudTrail log file validation is enabled (Config: CLOUD_TRAIL_LOG_FILE_VALIDATION_ENABLED).
- 3.4: Ensure CloudTrail trails are integrated with CloudWatch Logs.
- 3.7: Ensure CloudTrail logs encrypted at rest with KMS CMK.
- 4.1: Ensure a log metric filter and alarm exist for unauthorized API calls.
- 4.2: Ensure a log metric filter and alarm exist for Management Console sign-in without MFA.
- 4.3: Ensure a log metric filter and alarm exist for usage of root account.
- 4.4: Ensure a log metric filter and alarm exist for IAM policy changes.
- 4.5: Ensure a log metric filter and alarm exist for CloudTrail configuration changes.
- 4.6: Ensure a log metric filter and alarm exist for AWS Management Console authentication failures.
- 4.9: Ensure a log metric filter and alarm exist for VPC Security Group changes.
- 4.15: Ensure a log metric filter and alarm exist for AWS Organizations changes.

## 10.3 Incident Response Playbooks

A well-instrumented environment is only valuable if your team knows how to respond when an alarm fires. Incident response playbooks should be codified as Systems Manager Automation documents or Lambda functions, triggered by EventBridge rules matching finding patterns — reducing the time from detection to containment from hours to minutes.

### Playbook 1: Compromised IAM Credential

- Trigger: GuardDuty finding of type UnauthorizedAccess:IAMUser/InstanceCredentialExfiltration or similar.
- Step 1: Lambda attaches a deny-all inline policy to the IAM user/role within 30 seconds.
- Step 2: Lambda deactivates all access keys for the compromised principal.
- Step 3: Lambda queries CloudTrail Lake for all API calls by the principal in the last 24 hours.
- Step 4: SNS notification to the security team with the GuardDuty finding, credential deactivation confirmation, and CloudTrail activity summary.
- Step 5: ServiceNow/Jira ticket created with full context for post-incident investigation.

### Playbook 2: Public S3 Bucket Exposure

- Trigger: Config Rule NONCOMPLIANT event for S3_BUCKET_PUBLIC_READ_PROHIBITED or S3_ACCOUNT_LEVEL_PUBLIC_ACCESS_BLOCKS.
- Step 1: Lambda applies S3 Block Public Access at the bucket level immediately.
- Step 2: Lambda queries CloudTrail for the bucket policy change that caused the non-compliance.
- Step 3: Macie on-demand job triggered to scan the bucket for sensitive data.
- Step 4: Incident ticket created with bucket name, policy diff, Macie findings, and remediation confirmation.

### Playbook 3: EC2 Cryptominer Detection

- Trigger: GuardDuty finding CryptoCurrency:EC2/BitcoinTool.B or Impact:EC2/AbusedDomainRequest.Reputation.
- Step 1: Lambda quarantines the instance — modifies its Security Group to block all ingress/egress except SSM port.
- Step 2: Lambda creates an EBS snapshot of the instance's volumes for forensic analysis.
- Step 3: Lambda invokes EC2 Systems Manager to run a memory dump script via Run Command.
- Step 4: Security team notified with full context; automated forensic report generated.

## 10.4 Cost Optimization for Observability

Observability infrastructure can account for 10–30% of total AWS spend if not actively managed. The following optimization strategies consistently deliver the largest savings without sacrificing operational or compliance coverage:

| Area | Optimization Strategy | Typical Saving |
|---|---|---|
| CloudWatch Logs | Route high-volume, low-query logs (ALB, CloudFront, VPC Flow) to S3 Vended Logs; query with Athena | 60–80% |
| CloudTrail | Scope data events to sensitive resources only (not all S3 objects); use advanced event selectors | 50–70% |
| X-Ray / ADOT | Production sampling rate of 5%; 100% in staging; 0% for health/readiness check endpoints | 85–95% |
| GuardDuty | Add exclusion rules for known-safe IPs (internal scanners, load balancer health checks) | 10–30% |
| CloudWatch Metrics | Delete unused alarms and dashboards; use metric math instead of storing derived metrics | 5–20% |
| Config | Use periodic evaluation (not continuous) for resource types with infrequent changes | 15–25% |
| Security Hub | Create suppression rules for false-positive finding types after review | 10–20% |

| Area | Optimization Strategy | Typical Saving |
|---|---|---|
| Inspector | Scope to production ECR repositories; use agentless for dev accounts | 20–40% |

Table 10.2 — Observability cost optimization strategies.

## 10.5 Observability Maturity Model

Organizations adopt cloud observability in stages. The maturity model below provides a roadmap from basic compliance to proactive, AI-assisted security operations — each level building on the previous.

| Level | Name | Capabilities | Key Services |
|---|---|---|---|
| 1 | Foundational | CloudTrail on; basic CloudWatch alarms; GuardDuty enabled; S3 access logging; Config recorder running | CloudTrail, CloudWatch, GuardDuty, Config |
| 2 | Managed | Config Conformance Packs; Security Hub enabled; VPC Flow Logs org-wide; centralized logging; automated compliance reporting | Config, Security Hub, VPC Flow Logs, CloudWatch OAM |
| 3 | Proactive | Application tracing (ADOT); custom dashboards per team; automated remediation; incident playbooks; Macie sensitive data discovery | X-Ray/ADOT, EventBridge, SSM, Lambda, Macie |
| 4 | Optimized | Unified data store with Iceberg; AI anomaly detection; Security Agent; SBOM generation; predictive capacity planning | CloudWatch Unified Store, Inspector, Security Agent |
| 5 | Autonomous | AI-driven detection and automated containment; continuous compliance evidence generation; zero-touch telemetry for all new resources | Security Hub v2, Extended Threat Detection, Auto-enable |

Table 10.3 — Cloud observability and security maturity model (5 levels).

## 10.6 Future Directions

The trajectory of AWS observability and security is clear from the 2025–2026 release cycle: automation, unification, and AI-driven intelligence. Several trends will define the next phase:

- AI-native security: AWS Security Agent (preview, December 2025) signals AWS's intent to move from alerting to autonomous security action. Expect broader availability, integration with more test target types (APIs, infrastructure, IaC configurations), and tighter integration with the Security Hub risk model.
- OpenTelemetry convergence: The transition of X-Ray SDKs to maintenance mode marks the completion of AWS's pivot to open standards. Future CloudWatch features will be designed primarily for OTLP-format telemetry, and the ADOT Collector will become the standard telemetry collection agent for all AWS workloads.

- Apache Iceberg as the universal query layer: The CloudWatch unified data store materializing logs into S3 Tables (Iceberg) is a preview of a broader AWS strategy: all observability and security data will eventually be queryable via Apache Iceberg-compatible engines, eliminating proprietary query APIs and enabling multi-tool analysis.
- Zero-touch telemetry: The auto-enablement rules for CloudWatch (VPC flow logs in August 2025, CloudFront and Security Hub in April 2026) indicate a direction toward a 'compliant by default' model where any new AWS resource automatically emits the appropriate telemetry without manual configuration.
- AI-assisted investigation: Amazon Q Developer's security investigation capabilities, combined with CloudTrail Lake's natural-language query feature (preview), will make forensic investigation accessible to analysts without deep SQL expertise.

The organizations that will excel in cloud security and compliance over the next three to five years are those that build their observability architecture on open standards (OpenTelemetry, Apache Iceberg, OCSF), invest in automation from day one (Infrastructure as Code, automated remediation, automated evidence collection), and treat observability as a product with ongoing engineering investment rather than a one-time deployment.

CHAPTER Appendix A

# CloudWatch CLI & API Quick Reference

This appendix provides copy-paste ready AWS CLI commands and boto3 code snippets for the most common CloudWatch operations.

## A.1 Metrics

```
# List all namespaces in your account
aws cloudwatch list-metrics --output table

# List metrics for a specific namespace
aws cloudwatch list-metrics \
--namespace AWS/EC2 \
--dimensions Name=InstanceId,Value=i-0abc123def456

# Get metric statistics (last 1 hour, 5-minute periods)
aws cloudwatch get-metric-statistics \
--namespace AWS/EC2 \
--metric-name CPUUtilization \
--dimensions Name=InstanceId,Value=i-0abc123def456 \
--start-time $(date -u -d '1 hour ago' +%Y-%m-%dT%H:%M:%SZ) \
--end-time $(date -u +%Y-%m-%dT%H:%M:%SZ) \
--period 300 \
--statistics Average Maximum

# Publish a custom metric
aws cloudwatch put-metric-data \
--namespace MyApp/Business \
--metric-name OrdersProcessed \
--value 142 \
--unit Count \
--dimensions Name=Environment,Value=production

# boto3: Get metric data (preferred for large queries)
import boto3, datetime
cw = boto3.client('cloudwatch')
resp = cw.get_metric_data(
MetricDataQueries=[{
```

```
'Id': 'cpu', 'Label': 'CPU Utilization',
'MetricStat': {
'Metric': {
'Namespace': 'AWS/EC2',
'MetricName': 'CPUUtilization',
'Dimensions': [{'Name':'InstanceId','Value':'i-0abc123def456'}]
},
'Period': 300, 'Stat': 'Average'
}
}],
StartTime=datetime.datetime.utcnow() - datetime.timedelta(hours=1),
EndTime=datetime.datetime.utcnow()
)
```

## A.2 Alarms

```
# Create a CPU alarm with SNS notification
aws cloudwatch put-metric-alarm \
--alarm-name HighCPU-i-0abc123def456 \
--alarm-description 'EC2 CPU > 80% for 5 minutes' \
--namespace AWS/EC2 \
--metric-name CPUUtilization \
--dimensions Name=InstanceId,Value=i-0abc123def456 \
--period 300 \
--evaluation-periods 3 \
--threshold 80 \
--comparison-operator GreaterThanOrEqualToThreshold \
--statistic Average \
--alarm-actions arn:aws:sns:us-east-1:123456789012:ops-alerts \
--ok-actions arn:aws:sns:us-east-1:123456789012:ops-alerts \
--treat-missing-data breaching

# Create a composite alarm
aws cloudwatch put-composite-alarm \
--alarm-name WebTier-Degraded \
--alarm-rule 'ALARM(HighCPU) OR ALARM(High5xxRate)' \
--alarm-actions arn:aws:sns:us-east-1:123456789012:pagerduty

# List alarms in ALARM state
aws cloudwatch describe-alarms \
--state-value ALARM \
```

```
--query 'MetricAlarms[].{Name:AlarmName,Reason:StateReason}' \
--output table
```

```
# Suppress an alarm during maintenance
aws cloudwatch set-alarm-state \
--alarm-name HighCPU-i-0abc123def456 \
--state-value OK \
--state-reason 'Maintenance window'
```

## A.3 Log Groups, Streams & Insights

```
# Create a log group with 90-day retention
aws logs create-log-group \
--log-group-name /app/production/api
aws logs put-retention-policy \
--log-group-name /app/production/api \
--retention-in-days 90
```

```
# Tail logs in real time (CloudWatch equivalent)
aws logs tail /app/production/api --follow
```

```
# Start a CloudWatch Logs Insights query
aws logs start-query \
--log-group-name /app/production/api \
--start-time $(date -d '1 hour ago' +%s) \
--end-time $(date +%s) \
--query-string \
'fields @timestamp, @message
| filter @message like /ERROR/
| stats count(*) as errorCount by bin(5m)
| sort @timestamp desc
| limit 50'
```

```
# Get query results (poll until complete)
aws logs get-query-results --query-id <queryId>
```

```
# Enable Metric Filter for error counting
aws logs put-metric-filter \
--log-group-name /app/production/api \
--filter-name ErrorCount \
--filter-pattern '[timestamp, requestId, level=ERROR, ...]' \
--metric-transformations \
metricName=APIErrors,metricNamespace=MyApp,metricValue=1
```

## A.4 Dashboards

```
# Create dashboard from JSON body file
aws cloudwatch put-dashboard \
--dashboard-name ProductionOverview \
--dashboard-body file://dashboard.json

# Minimal dashboard JSON structure
# {
# "widgets": [
# {
# "type": "metric",
# "x": 0, "y": 0, "width": 12, "height": 6,
# "properties": {
# "metrics": [["AWS/EC2","CPUUtilization","InstanceId","i-0abc"]],
# "view": "timeSeries", "stat": "Average",
# "period": 300, "title": "EC2 CPU"
# }
# }
# ]
# }

# List dashboards
aws cloudwatch list-dashboards --output table

# Share a dashboard (read-only public link)
aws cloudwatch put-dashboard-sharing-policy \
--dashboard-name ProductionOverview \
--sharing-policy '{"Type":"PUBLIC"}'
```

CHAPTER Appendix B

# CloudTrail Event Anatomy

Understanding the structure of a CloudTrail event is essential for writing effective queries, detection rules, and forensic investigations.

## B.1 Full Management Event Example


```
{
"eventVersion": "1.09",
"userIdentity": {
"type": "AssumedRole",
"principalId": "AROAEXAMPLEID:session-name",
"arn": "arn:aws:sts::123456789012:assumed-role/MyRole/session",
"accountId": "123456789012",
"sessionContext": {
"sessionIssuer": {
"type": "Role",
"arn": "arn:aws:iam::123456789012:role/MyRole",
"accountId": "123456789012",
"userName": "MyRole"
},
"attributes": {
"creationDate": "2026-04-15T00:00:00Z",
"mfaAuthenticated": "false"
}
}
},
"eventTime": "2026-04-15T02:00:00Z",
"eventSource": "s3.amazonaws.com",
"eventName": "GetObject",
"awsRegion": "us-east-1",
"sourceIPAddress": "203.0.113.42",
"userAgent": "aws-cli/2.15.0 Python/3.12.0 ...",
"requestParameters": {
"bucketName": "my-sensitive-bucket",
```

```
"key": "finance/payroll-2026-Q1.csv"
},
"responseElements": null,
"requestID": "EXAMPLE123456789",
"eventID": "a1b2c3d4-e5f6-7890-abcd-ef1234567890",
"readOnly": true,
"resources": [{
"ARN": "arn:aws:s3:::my-sensitive-bucket/finance/payroll-2026-Q1.csv",
"accountId": "123456789012",
"type": "AWS::S3::Object"
}],
"eventType": "AwsApiCall",
"managementEvent": false,
"recipientAccountId": "123456789012"
}
```


## B.2 userIdentity Types Reference

| Type | When Used | Key Fields |
|---|---|---|
| Root | Direct root account login | accountId (no userName) |
| IAMUser | Long-term IAM user credentials | userName, arn |
| AssumedRole | STS AssumeRole (most common) | principalId, sessionIssuer |
| FederatedUser | SAML/OIDC federation | principalId, federatedIdentity |
| AWSService | Service-to-service calls | invokedBy field populated |
| AWSAccount | Cross-account access | accountId of calling account |
| WebIdentityUser | Cognito/OIDC web identity | principalId, identityProvider |

Table B.1 — CloudTrail userIdentity types and when each appears.

## B.3 High-Value Events to Monitor

| Event Source | Event Name | Why It Matters |
|---|---|---|
| iam.amazonaws.com | CreateUser / AttachUserPolicy | Privilege escalation |
| iam.amazonaws.com | CreateAccessKey | New programmatic credentials |
| iam.amazonaws.com | UpdateAssumeRolePolicy | Trust policy modification |
| sts.amazonaws.com | AssumeRoleWithWebIdentity | OIDC token exchange |
| signin.amazonaws.com | ConsoleLogin (errorCode) | Failed console login |
| kms.amazonaws.com | DisableKey / ScheduleKeyDeletion | Key destruction risk |
| ec2.amazonaws.com | AuthorizeSecurityGroupIngress | Firewall rule change |
| s3.amazonaws.com | PutBucketPublicAccessBlock=false | Bucket made public |
| cloudtrail.amazonaws.com | StopLogging / DeleteTrail | Audit evasion |
| guardduty.amazonaws.com | DisassociateFromMasterAccount | Security evasion |
| organizations.amazonaws.com | LeaveOrganization | SCP bypass attempt |

Table B.2 — High-value CloudTrail events for detection engineering.

## B.4 CloudTrail Lake Athena DDL

```
-- Create external table for CloudTrail S3 logs (Athena)
CREATE EXTERNAL TABLE cloudtrail_logs (
eventVersion STRING,
userIdentity STRUCT<
type: STRING,
principalId: STRING,
arn: STRING,
accountId: STRING,
userName: STRING,
sessionContext: STRUCT<
sessionIssuer: STRUCT<
type: STRING,
userName: STRING,
arn: STRING
>
>
```

```
>,
eventTime STRING,
eventSource STRING,
eventName STRING,
awsRegion STRING,
sourceIPAddress STRING,
userAgent STRING,
errorCode STRING,
errorMessage STRING,
requestParameters STRING,
responseElements STRING,
eventID STRING,
eventType STRING,
readOnly BOOLEAN,
resources ARRAY<STRUCT<ARN:STRING, accountId:STRING, type:STRING>>
)
PARTITIONED BY (region STRING, year STRING, month STRING, day STRING)
ROW FORMAT SERDE 'com.amazon.emr.hive.serde.CloudTrailSerde'
STORED AS INPUTFORMAT 'com.amazon.emr.cloudtrail.CloudTrailInputFormat'
OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION 's3://my-cloudtrail-bucket/AWSLogs/123456789012/CloudTrail/';

-- Repair partitions after new data arrives
MSCK REPAIR TABLE cloudtrail_logs;

-- Find all IAM key creations in the last 7 days
SELECT eventTime, userIdentity.userName, requestParameters
FROM cloudtrail_logs
WHERE eventName = 'CreateAccessKey'
AND year = '2026' AND month = '04'
ORDER BY eventTime DESC;
```

CHAPTER Appendix C

# GuardDuty & Security Hub Automation

Automated response to GuardDuty findings and Security Hub controls reduces mean-time-to-remediate (MTTR) and eliminates repetitive manual tasks. This appendix provides ready-to-deploy EventBridge rules and Lambda functions.

## C.1 EventBridge Rule — Route GuardDuty Findings to Lambda

```
# Terraform: EventBridge rule for HIGH/CRITICAL GuardDuty findings
resource "aws_cloudwatch_event_rule" "gd_high_findings" {
name = "guardduty-high-critical"
description = "Route HIGH and CRITICAL GuardDuty findings to response Lambda"
event_pattern = jsonencode({
source = ["aws.guardduty"],
detail-type = ["GuardDuty Finding"],
detail = {
severity = [{ numeric = [">=", 7] }]
}
})
}

resource "aws_cloudwatch_event_target" "gd_lambda" {
rule = aws_cloudwatch_event_rule.gd_high_findings.name
arn = aws_lambda_function.gd_responder.arn
}
```

## C.2 Lambda Response Function (Python)

```
import boto3, json, os

ec2 = boto3.client('ec2')
sns = boto3.client('sns')
TOPIC = os.environ['ALERT_TOPIC_ARN']

def handler(event, context):
detail = event['detail']
severity = detail['severity']
ftype = detail['type']
```

```
account = detail['accountId']
region = detail['region']

# Extract compromised instance (if applicable)
instance_id = None
try:
instance_id = detail['resource']['instanceDetails']['instanceId']
except (KeyError, TypeError):
pass

msg = f'GuardDuty [{severity:.1f}] {ftype}\n'
msg += f'Account: {account} | Region: {region}\n'

if instance_id and severity >= 8.0:
# Auto-isolate: remove all security groups, add quarantine SG
quarantine_sg = os.environ.get('QUARANTINE_SG_ID')
if quarantine_sg:
ec2.modify_instance_attribute(
InstanceId=instance_id,
Groups=[quarantine_sg]
)
msg += f'ACTION: Instance {instance_id} ISOLATED\n'

# Always send SNS notification
sns.publish(
TopicArn=TOPIC,
Subject=f'GuardDuty Alert — Severity {severity:.1f}',
Message=msg
)
return {'statusCode': 200}
```

## C.3 Security Hub — Bulk Finding Status Update (CLI)

```
# Suppress all INFORMATIONAL findings in Security Hub (bulk)
aws securityhub batch-update-findings \
--finding-identifiers \
Id=arn:aws:securityhub:us-east-1:123456789012:finding/abc123,\
ProductArn=arn:aws:securityhub:us-east-1::product/aws/securityhub \
--workflow Status=SUPPRESSED

# List all FAILED CIS controls (Security Hub)
aws securityhub get-findings \
--filters '{
```

```
"ComplianceStatus": [{"Value": "FAILED", "Comparison": "EQUALS"}],
```

```
"GeneratorId": [{"Value": "arn:aws:securityhub:::ruleset/cis-aws-", "Comparison":
"PREFIX"}]
}' \
--query 'Findings[].{Title:Title,Severity:Severity.Label,Account:AwsAccountId}' \
--output table

# Get Security Hub standards summary
aws securityhub describe-standards-controls \
--standards-subscription-arn \
arn:aws:securityhub:us-east-1:123456789012:subscription/cis-aws-foundations-benchmark/v/1.4
.0 \
--query 'Controls[?ControlStatus==`FAILED`].{ID:ControlId,Title:Title}' \
--output table
```

CHAPTER Appendix D

# CIS Benchmark CloudWatch Metric Filters

CIS AWS Foundations Benchmark v1.4 Section 4 requires 15 CloudWatch metric filters and alarms. The Terraform below provisions all 15 against a CloudTrail log group.

```
variable "log_group_name" {
default = "/aws/cloudtrail/organization"
}
variable "alarm_sns_arn" {}

locals {
cis_filters = {
"cis-3-1-unauth-api" = "{ ($.errorCode = \"*UnauthorizedOperation\") || ($.errorCode = \"AccessDenied*\") }"
"cis-3-2-no-mfa-console" = "{ ($.eventName = \"ConsoleLogin\") && ($.additionalEventData.MFAUsed != \"Yes\") && ($.userIdentity.type = \"IAMUser\") && ($.responseElements.ConsoleLogin = \"Success\") }"
"cis-3-3-root-usage" = "{ $.userIdentity.type = \"Root\" && $.userIdentity.invokedBy NOT EXISTS && $.eventType != \"AwsServiceEvent\" }"
"cis-3-4-iam-policy" = "{ ($.eventName = \"DeleteGroupPolicy\") || ($.eventName = \"DeleteRolePolicy\") || ($.eventName = \"DeleteUserPolicy\") || ($.eventName = \"PutGroupPolicy\") || ($.eventName = \"PutRolePolicy\") || ($.eventName = \"PutUserPolicy\") || ($.eventName = \"CreatePolicy\") || ($.eventName = \"DeletePolicy\") || ($.eventName = \"CreatePolicyVersion\") || ($.eventName = \"DeletePolicyVersion\") || ($.eventName = \"SetDefaultPolicyVersion\") || ($.eventName = \"AttachRolePolicy\") || ($.eventName = \"DetachRolePolicy\") || ($.eventName = \"AttachUserPolicy\") || ($.eventName = \"DetachUserPolicy\") || ($.eventName = \"AttachGroupPolicy\") || ($.eventName = \"DetachGroupPolicy\") }"
"cis-3-5-cloudtrail" = "{ ($.eventName = \"CreateTrail\") || ($.eventName = \"UpdateTrail\") || ($.eventName = \"DeleteTrail\") || ($.eventName = \"StartLogging\") || ($.eventName = \"StopLogging\") }"
"cis-3-6-console-auth" = "{ ($.eventName = \"ConsoleLogin\") && ($.errorMessage = \"Failed authentication\") }"
"cis-3-7-cmk" = "{ ($.eventSource = \"kms.amazonaws.com\") && (($.eventName = \"DisableKey\") || ($.eventName = \"ScheduleKeyDeletion\")) }"
"cis-3-8-s3-bucket" = "{ ($.eventSource = \"s3.amazonaws.com\") && (($.eventName = \"PutBucketAcl\") || ($.eventName = \"PutBucketPolicy\") || ($.eventName = \"PutBucketCors\") || ($.eventName = \"PutBucketLifecycle\") || ($.eventName = \"PutBucketReplication\") || ($.eventName = \"DeleteBucketPolicy\") || ($.eventName = \"DeleteBucketCors\") || ($.eventName = \"DeleteBucketLifecycle\") || ($.eventName = \"DeleteBucketReplication\")) }"
```

```
"cis-3-9-config" = "{ ($.eventSource = \"config.amazonaws.com\") && (($.eventName =
\"StopConfigurationRecorder\") || ($.eventName = \"DeleteDeliveryChannel\") || ($.eventName
= \"PutDeliveryChannel\") || ($.eventName = \"PutConfigurationRecorder\")) }"

"cis-3-10-secgroup" = "{ ($.eventName = \"AuthorizeSecurityGroupIngress\") || ($.eventName =
\"AuthorizeSecurityGroupEgress\") || ($.eventName = \"RevokeSecurityGroupIngress\") ||
($.eventName = \"RevokeSecurityGroupEgress\") || ($.eventName = \"CreateSecurityGroup\") ||
($.eventName = \"DeleteSecurityGroup\") }"

"cis-3-11-nacl" = "{ ($.eventName = \"CreateNetworkAcl\") || ($.eventName =
\"CreateNetworkAclEntry\") || ($.eventName = \"DeleteNetworkAcl\") || ($.eventName =
\"DeleteNetworkAclEntry\") || ($.eventName = \"ReplaceNetworkAclEntry\") || ($.eventName =
\"ReplaceNetworkAclAssociation\") }"

"cis-3-12-gateway" = "{ ($.eventName = \"CreateCustomerGateway\") || ($.eventName =
\"DeleteCustomerGateway\") || ($.eventName = \"AttachInternetGateway\") || ($.eventName =
\"CreateInternetGateway\") || ($.eventName = \"DeleteInternetGateway\") || ($.eventName =
\"DetachInternetGateway\") }"

"cis-3-13-routetable" = "{ ($.eventName = \"CreateRoute\") || ($.eventName =
\"CreateRouteTable\") || ($.eventName = \"ReplaceRoute\") || ($.eventName =
\"ReplaceRouteTableAssociation\") || ($.eventName = \"DeleteRouteTable\") || ($.eventName =
\"DeleteRoute\") || ($.eventName = \"DisassociateRouteTable\") }"

"cis-3-14-vpc" = "{ ($.eventName = \"CreateVpc\") || ($.eventName = \"DeleteVpc\") ||
($.eventName = \"ModifyVpcAttribute\") || ($.eventName = \"AcceptVpcPeeringConnection\") ||
($.eventName = \"CreateVpcPeeringConnection\") || ($.eventName =
\"DeleteVpcPeeringConnection\") || ($.eventName = \"RejectVpcPeeringConnection\") ||
($.eventName = \"AttachClassicLinkVpc\") || ($.eventName = \"DetachClassicLinkVpc\") ||
($.eventName = \"DisableVpcClassicLink\") || ($.eventName = \"EnableVpcClassicLink\") }"

"cis-3-15-org" = "{ ($.eventSource = \"organizations.amazonaws.com\") && (($.eventName =
\"AcceptHandshake\") || ($.eventName = \"AttachPolicy\") || ($.eventName =
\"CreateAccount\") || ($.eventName = \"CreateOrganizationalUnit\") || ($.eventName =
\"CreatePolicy\") || ($.eventName = \"DeclineHandshake\") || ($.eventName =
\"DeleteOrganizationalUnit\") || ($.eventName = \"DeletePolicy\") || ($.eventName =
\"DetachPolicy\") || ($.eventName = \"DisablePolicyType\") || ($.eventName =
\"EnablePolicyType\") || ($.eventName = \"InviteAccountToOrganization\") || ($.eventName =
\"LeaveOrganization\") || ($.eventName = \"MoveAccount\") || ($.eventName =
\"RemoveAccountFromOrganization\") || ($.eventName = \"UpdateOrganizationalUnit\") ||
($.eventName = \"UpdatePolicy\")) }"

}

}
```

```
resource "aws_cloudwatch_log_metric_filter" "cis" {

for_each = local.cis_filters

name = each.key

log_group_name = var.log_group_name

pattern = each.value

metric_transformation {

name = each.key

namespace = "CISBenchmark"

value = "1"
```

```
}
}
```

```
resource "aws_cloudwatch_metric_alarm" "cis" {
for_each = local.cis_filters
alarm_name = "${each.key}-alarm"
metric_name = each.key
namespace = "CISBenchmark"
period = "300"
evaluation_periods = "1"
threshold = "1"
comparison_operator = "GreaterThanOrEqualToThreshold"
statistic = "Sum"
alarm_actions = [var.alarm_sns_arn]
treat_missing_data = "notBreaching"
}
```

CHAPTER Appendix E

# AWS Config Conformance Pack YAML (SOC 2)

The following AWS Config Conformance Pack implements a custom SOC 2 Type II control mapping using managed Config rules. Deploy via the AWS Console, CLI, or AWS Organizations for org-wide coverage.

```
# soc2-conformance-pack.yaml
Parameters:
MaxAccessKeyAge:
Type: String
Default: '90'
MaxPasswordAge:
Type: String
Default: '90'
MinPasswordLength:
Type: String
Default: '14'

Resources:

# CC6.1 — Logical access controls
MFAEnabledForConsoleAccess:
Type: AWS::Config::ConfigRule
Properties:
ConfigRuleName: soc2-mfa-console-access
Source:
Owner: AWS
SourceIdentifier: MFA_ENABLED_FOR_IAM_CONSOLE_ACCESS

RootMFAEnabled:
Type: AWS::Config::ConfigRule
Properties:
ConfigRuleName: soc2-root-mfa
Source:
Owner: AWS
SourceIdentifier: ROOT_MFA_ENABLED

AccessKeyRotation:
Type: AWS::Config::ConfigRule
```

```
    Properties:
      ConfigRuleName: soc2-access-key-rotation
      Source:
        Owner: AWS
        SourceIdentifier: ACCESS_KEYS_ROTATED
      InputParameters:
        maxAccessKeyAge: !Ref MaxAccessKeyAge

  # CC6.6 — Encryption in transit
  ELBTLSHTTPSOnly:
    Type: AWS::Config::ConfigRule
    Properties:
      ConfigRuleName: soc2-elb-https
      Source:
        Owner: AWS
        SourceIdentifier: ALARMS_HTTPS_CHECK

  S3BucketSSLOnly:
    Type: AWS::Config::ConfigRule
    Properties:
      ConfigRuleName: soc2-s3-ssl
      Source:
        Owner: AWS
        SourceIdentifier: S3_BUCKET_SSL_REQUESTS_ONLY

  # CC6.7 — Encryption at rest
  S3BucketEncrypted:
    Type: AWS::Config::ConfigRule
    Properties:
      ConfigRuleName: soc2-s3-encryption
      Source:
        Owner: AWS
        SourceIdentifier: S3_BUCKET_SERVER_SIDE_ENCRYPTION_ENABLED

  EBSEncrypted:
    Type: AWS::Config::ConfigRule
    Properties:
      ConfigRuleName: soc2-ebs-encryption
      Source:
        Owner: AWS
        SourceIdentifier: EC2_EBS_ENCRYPTION_BY_DEFAULT

  RDSEncrypted:
```

```
    Type: AWS::Config::ConfigRule
    Properties:
      ConfigRuleName: soc2-rds-encryption
      Source:
        Owner: AWS
        SourceIdentifier: RDS_STORAGE_ENCRYPTED

  # CC7.2 — Logging and monitoring
  CloudTrailEnabled:
    Type: AWS::Config::ConfigRule
    Properties:
      ConfigRuleName: soc2-cloudtrail-enabled
      Source:
        Owner: AWS
        SourceIdentifier: CLOUD_TRAIL_ENABLED

  CloudTrailLogValidation:
    Type: AWS::Config::ConfigRule
    Properties:
      ConfigRuleName: soc2-cloudtrail-validation
      Source:
        Owner: AWS
        SourceIdentifier: CLOUD_TRAIL_LOG_FILE_VALIDATION_ENABLED

  GuardDutyEnabled:
    Type: AWS::Config::ConfigRule
    Properties:
      ConfigRuleName: soc2-guardduty
      Source:
        Owner: AWS
        SourceIdentifier: GUARDDUTY_ENABLED_CENTRALIZED

  SecurityHubEnabled:
    Type: AWS::Config::ConfigRule
    Properties:
      ConfigRuleName: soc2-security-hub
      Source:
        Owner: AWS
        SourceIdentifier: SECURITYHUB_ENABLED

  # CC8.1 — Change management
  CloudFormationStackDriftDetection:
    Type: AWS::Config::ConfigRule
```

```
Properties:
ConfigRuleName: soc2-cfn-drift
Source:
Owner: AWS
SourceIdentifier: CLOUDFORMATION_STACK_DRIFT_DETECTION_CHECK

# Deploy via CLI:
# aws configservice put-conformance-pack \
# --conformance-pack-name SOC2-Custom \
# --template-body file://soc2-conformance-pack.yaml
```

CHAPTER Appendix F

# Multi-Account Architecture Reference

This appendix provides a reference account structure, Service Control Policy (SCP) examples, and CLI commands for configuring delegated administrators in AWS Organizations.

## F.1 Recommended Account Structure

| Account | Purpose | Key Services |
|---|---|---|
| Management (Root) | Org management only; no workloads | Organizations, SCPs, IAM Identity Center |
| Log Archive | Immutable centralized log storage | S3 (Object Lock), CloudTrail, Config |
| Security Tooling | Security services hub | GuardDuty admin, Security Hub admin, Inspector admin |
| Network Hub | Shared network resources | Transit Gateway, Route 53, Direct Connect, VPC |
| Shared Services | Common infrastructure | AMI pipelines, CodeArtifact, Service Catalog |
| Prod Workloads | Production applications | Application-specific services, tightly scoped IAM |
| Dev/Test | Development and testing | Broader permissions, shorter log retention |
| Sandbox | Experimentation only | SCPs prevent data exfiltration; no prod access |

Table F.1 — Recommended AWS Organizations account structure for enterprise environments.

## F.2 SCP — Prevent CloudTrail Disable

```
{
"Version": "2012-10-17",
"Statement": [
{
"Sid": "DenyCloudTrailDisable",
"Effect": "Deny",
"Action": [
"cloudtrail:StopLogging",
"cloudtrail:DeleteTrail",
```

```
"cloudtrail:UpdateTrail",
"cloudtrail:PutEventSelectors"
],
"Resource": "*",
"Condition": {
"StringNotLike": {
"aws:PrincipalArn": [
"arn:aws:iam::*:role/CloudTrailAdmin",
"arn:aws:iam::*:role/BreakGlassRole"
]
}
}
},
{
"Sid": "DenyGuardDutyDisable",
"Effect": "Deny",
"Action": [
"guardduty:DeleteDetector",
"guardduty:DisassociateFromMasterAccount",
"guardduty:StopMonitoringMembers",
"guardduty:UpdateDetector"
],
"Resource": "*",
"Condition": {
"StringNotLike": {
"aws:PrincipalArn": "arn:aws:iam::*:role/SecurityAdmin"
}
}
}
]
}
```

## F.3 Delegated Administrator CLI Reference

```
# Register delegated admin for Security Hub
aws organizations enable-aws-service-principal \
--service-principal securityhub.amazonaws.com
aws securityhub enable-organization-admin-account \
--admin-account-id 222222222222
```

```
# Register delegated admin for GuardDuty
aws organizations enable-aws-service-principal \
--service-principal guardduty.amazonaws.com
aws guardduty enable-organization-admin-account \
--admin-account-id 222222222222

# Register delegated admin for CloudTrail
aws cloudtrail register-organization-delegated-admin \
--member-account-id 111111111111

# Register delegated admin for AWS Config
aws organizations register-delegated-administrator \
--account-id 111111111111 \
--service-principal config-multiaccountsetup.amazonaws.com

# List all delegated administrators
aws organizations list-delegated-administrators \
--query 'DelegatedAdministrators[].{Account:Id,Service:ServicePrincipal}' \
--output table
```

CHAPTER Appendix G

# Incident Response Quick Reference

This appendix provides CLI command sequences for common AWS security incident response scenarios. All commands are non-destructive (read-only or defensive) unless explicitly labeled as remediation actions.

## G.1 Initial Triage Commands

```
# Identify who called what in the last hour (CloudTrail)
aws logs start-query \
--log-group-name /aws/cloudtrail/organization \
--start-time $(date -d '1 hour ago' +%s) \
--end-time $(date +%s) \
--query-string 'fields eventTime, userIdentity.arn, eventName, sourceIPAddress
| sort @timestamp desc | limit 200'

# List all active access keys for a user
aws iam list-access-keys --user-name <username>

# Get console login history from CloudTrail
aws cloudtrail lookup-events \
--lookup-attributes AttributeKey=EventName,AttributeValue=ConsoleLogin \
--start-time $(date -d '24 hours ago' --iso-8601=seconds) \
--query 'Events[].{Time:EventTime,User:Username,IP:CloudTrailEvent}' \
--output table

# Check for new/modified IAM entities in the last 24h
aws iam generate-credential-report
aws iam get-credential-report \
--query 'Content' --output text | base64 -d | column -t -s,

# List open GuardDuty findings (last 24h)
aws guardduty list-findings \
--detector-id $(aws guardduty list-detectors --query 'DetectorIds[0]' --output text) \
--finding-criteria \
'{"Criterion":{"updatedAt":{"GreaterThanOrEqual":['$(date -d '24 hours ago' +%s000)']}}}'
```

## G.2 EC2 Instance Isolation (Containment)

```
# STEP 1: Capture current security groups (for rollback)
INSTANCE_ID=i-0abc123def456
ORIG_SGS=$(aws ec2 describe-instances \
--instance-ids $INSTANCE_ID \
--query 'Reservations[0].Instances[0].SecurityGroups[].GroupId' \
--output text)
echo "Original SGS: $ORIG_SGS" | tee incident-$INSTANCE_ID.log

# STEP 2: Create quarantine security group (no inbound, no outbound)
VPC_ID=$(aws ec2 describe-instances \
--instance-ids $INSTANCE_ID \
--query 'Reservations[0].Instances[0].VpcId' --output text)
QUARANTINE_SG=$(aws ec2 create-security-group \
--group-name quarantine-$(date +%s) \
--description 'Incident quarantine — no inbound or outbound' \
--vpc-id $VPC_ID \
--query 'GroupId' --output text)

# Remove default egress rule
aws ec2 revoke-security-group-egress \
--group-id $QUARANTINE_SG \
--ip-permissions '[{"IpProtocol":"-1","IpRanges":[{"CidrIp":"0.0.0.0/0"}]}]'

# STEP 3: Apply quarantine SG (REMEDIATION — isolates instance)
aws ec2 modify-instance-attribute \
--instance-id $INSTANCE_ID \
--groups $QUARANTINE_SG

# STEP 4: Disable termination protection (allow safe shutdown)
aws ec2 modify-instance-attribute \
--instance-id $INSTANCE_ID \
--no-disable-api-termination
```

## G.3 Evidence Collection

```
# Create forensic EBS snapshot from running instance
VOLUME_ID=$(aws ec2 describe-instances \
--instance-ids $INSTANCE_ID \
--query 'Reservations[0].Instances[0].BlockDeviceMappings[0].Ebs.VolumeId' \
--output text)
```

```
aws ec2 create-snapshot \
```

```
--volume-id $VOLUME_ID \
--description "Incident $(date +%Y%m%d) — forensic snapshot of $INSTANCE_ID" \
--tag-specifications \
'ResourceType=snapshot,Tags=[{Key=incident,Value=IR-2026-001},{Key=chain-of-custody,Value=preserved}]'

# Dump CloudTrail events for specific principal (last 7 days)
PRINCIPAL_ARN='arn:aws:sts::123456789012:assumed-role/MyRole/session'
aws cloudtrail lookup-events \
--lookup-attributes AttributeKey=Username,AttributeValue="$PRINCIPAL_ARN" \
--start-time $(date -d '7 days ago' --iso-8601=seconds) \
--output json > evidence-cloudtrail-$(date +%Y%m%d).json

# Disable compromised access key (REMEDIATION)
aws iam update-access-key \
--user-name <username> \
--access-key-id AKIAIOSFODNN7EXAMPLE \
--status Inactive

# Revoke all active sessions for a role
aws iam put-role-policy \
--role-name CompromisedRole \
--policy-name RevokeAllSessions \
--policy-document \
'{"Version":"2012-10-17","Statement":[{"Effect":"Deny",
"Action":"*","Resource":"*","Condition":
{"DateLessThan":{"aws:TokenIssueTime":"'$(date -u +%Y-%m-%dT%H:%M:%SZ)'"}}}]}'
```

CHAPTER Appendix H

# Glossary of Terms and Abbreviations

Key terms, acronyms, and abbreviations used throughout this book.

| Term / Acronym | Definition |
|---|---|
| ADOT | AWS Distro for OpenTelemetry — AWS's production-ready, enterprise-supported distribution of the OpenTelemetry Collector and SDKs. |
| ASFF | Amazon Security Finding Format — the JSON schema used by Security Hub to normalize findings from all integrated security services. |
| ATM | AWS Trusted Advisor — proactive service recommending best practices across cost, performance, security, fault tolerance, and service limits. |
| CSPM | Cloud Security Posture Management — tools and practices that continuously assess cloud configurations against security benchmarks. |
| CIS | Center for Internet Security — non-profit that publishes security benchmarks including the CIS AWS Foundations Benchmark. |
| CSIRT | Computer Security Incident Response Team — the team or function responsible for detecting, analyzing, and responding to security incidents. |
| CTL | CloudTrail Lake — AWS managed data lake for storing, querying (via SQL), and federating CloudTrail event data. |
| DAST | Dynamic Application Security Testing — security testing that evaluates a running application from the outside for vulnerabilities. |
| DMARC | Domain-based Message Authentication, Reporting and Conformance — email authentication protocol, relevant to SES logging and monitoring. |
| DORA | Digital Operational Resilience Act — EU regulation (effective Jan 2025) requiring financial entities to demonstrate ICT risk management. |
| EMF | Embedded Metrics Format — CloudWatch protocol for embedding metric data inside structured log lines. |
| ETA | Extended Threat Detection — GuardDuty feature (GA Dec 2025) correlating sequences of events across services into multi-stage attack findings. |
| FedRAMP | Federal Risk and Authorization Management Program — US federal framework for cloud service security authorization. |
| GDPR | General Data Protection Regulation — EU data privacy regulation with logging and breach notification implications for AWS workloads. |
| HIPAA | Health Insurance Portability and Accountability Act — US regulation governing protected health information (PHI) in AWS workloads. |

| Term / Acronym | Definition |
|---|---|
| IaC | Infrastructure as Code — managing cloud resources through code (Terraform, CloudFormation, CDK) for reproducibility and audit. |
| IMDSv2 | Instance Metadata Service version 2 — session-oriented EC2 metadata access that prevents SSRF-based credential theft. |
| KMS | AWS Key Management Service — managed service for creating and controlling cryptographic keys used for data encryption. |
| Lake Formation | AWS service for building, securing, and managing data lakes on Amazon S3. |
| MACIE | Amazon Macie — ML-powered service that discovers and protects sensitive data in S3. |
| MTTD | Mean Time to Detect — average time between a security event occurring and its detection by monitoring tools. |
| MTTR | Mean Time to Remediate — average time from incident detection to full resolution and recovery. |
| OCSF | Open Cybersecurity Schema Framework — vendor-neutral security event schema adopted by AWS for Security Lake and other services. |
| OTLP | OpenTelemetry Protocol — the wire protocol for transmitting telemetry data (traces, metrics, logs) between ADOT and backends. |
| PCI DSS | Payment Card Industry Data Security Standard — requirements for organizations handling credit card data, applicable to AWS workloads. |
| SAIF | AWS Shared AI Responsibility Framework — framework for understanding security responsibilities for AI/ML workloads on AWS. |
| SBOM | Software Bill of Materials — catalog of all software components in an application, used by Inspector for vulnerability analysis. |
| SCP | Service Control Policy — AWS Organizations policy that restricts IAM permissions across entire OUs or accounts. |
| SIEM | Security Information and Event Management — platforms (Splunk, Microsoft Sentinel) that aggregate and correlate security logs. |
| SOC 2 | Service Organization Control Type 2 — audit framework assessing security, availability, processing integrity, confidentiality, and privacy. |
| SSRF | Server-Side Request Forgery — attack type where a server is manipulated to make requests on behalf of an attacker, often targeting IMDS. |
| STS | AWS Security Token Service — service that vends temporary, limited-privilege credentials for IAM roles. |
| VPC | Virtual Private Cloud — logically isolated section of AWS cloud infrastructure with its own networking configuration. |
| WAF | AWS Web Application Firewall — managed service for filtering malicious HTTP/S traffic to CloudFront, ALB, API Gateway, or AppSync. |

| Term / Acronym | Definition |
|---|---|
| XDR | Extended Detection and Response — security approach integrating telemetry across endpoints, network, and cloud for unified threat detection. |

Table H.1 — Glossary of terms and acronyms used in this book.

## H.2 AWS Monitoring Service Limits Reference

Service quotas (formerly called limits) control the maximum number of resources you can create in each AWS account and region. Monitoring services have their own quotas which may require increase requests via the Service Quotas console for large enterprise deployments.

| Service | Quota Name | Default Limit | Adjustable |
|---|---|---|---|
| CloudWatch | Metrics per account | 10,000,000 custom metrics | Yes |
| CloudWatch | Alarms per account | 5,000 standard alarms | Yes |
| CloudWatch | Dashboards per account | 500 | Yes |
| CloudWatch | Metric filters per log group | 1,000 | No |
| CloudWatch | Log groups per account | 1,000,000 | No |
| CloudWatch | Logs Insights concurrent queries | 10 per account | Yes |
| CloudWatch | Contributor Insights rules | 200 | No |
| CloudTrail | Trails per region | 5 | Yes |
| CloudTrail | CloudTrail Lake event data stores | 5 per account | Yes |
| CloudTrail | CloudTrail Lake queries (concurrent) | 10 | No |
| GuardDuty | Findings per account (active) | 1,000,000 | No |
| GuardDuty | Suppression rules | 100 | No |
| Security Hub | Findings per account (active) | 10,000,000 | No |
| Security Hub | Custom insights | 100 | No |
| Security Hub | Automation rules | 100 | No |
| Config | Config rules per account | 500 | Yes |
| Config | Conformance packs per account | 50 | Yes |
| Config | Remediation configurations | 500 | No |
| Inspector | Active findings per account | 1,000,000 | No |
| X-Ray / ADOT | Groups per account | 25 | No |

| Service | Quota Name | Default Limit | Adjustable |
|---|---|---|---|
| X-Ray / ADOT | Sampling rules per account | 25 (default group + 24 custom) | No |

Table H.2 — Default service quotas for AWS monitoring services (April 2026). Request increases via aws service-quotas request-service-quota-increase.

## H.3 AWS Monitoring API Endpoint Reference

All AWS monitoring service APIs use HTTPS endpoints following the format: ..amazonaws.com. The table below lists the service endpoint prefix for each major monitoring service, which is also the value used in the --endpoint-url CLI flag for testing with mock endpoints in development.

| Service | CLI Service Name | API Endpoint Prefix | Global Endpoint? |
|---|---|---|---|
| Amazon CloudWatch | cloudwatch | monitoring | No (regional) |
| CloudWatch Logs | logs | logs | No (regional) |
| CloudWatch Synthetics | synthetics | synthetics | No (regional) |
| AWS CloudTrail | cloudtrail | cloudtrail | No (regional) |
| AWS Config | configservice | config | No (regional) |
| Amazon GuardDuty | guardduty | guardduty | No (regional) |
| AWS Security Hub | securityhub | securityhub | No (regional) |
| Amazon Inspector | inspector2 | inspector2 | No (regional) |
| Amazon Macie | macie2 | macie2 | No (regional) |
| AWS X-Ray | xray | xray | No (regional) |
| Amazon Security Lake | securitylake | securitylake | No (regional) |
| AWS Organizations | organizations | organizations | Yes (us-east-1 only) |
| AWS IAM | iam | iam | Yes (global) |
| AWS STS | sts | sts | Yes (global, regional preferred) |

Table H.3 — AWS monitoring service API endpoint reference. Regional endpoints follow the pattern: https://..amazonaws.com.

## H.4 CloudWatch Container Insights Metrics Reference

Container Insights collects a rich set of metrics for Amazon ECS, EKS, and Kubernetes on EC2. The following table is a reference for the most operationally significant metrics available in the ContainerInsights namespace.

| Metric Name | Dimension(s) | Unit | Description |
|---|---|---|---|
| node_cpu_utilization | ClusterName, NodeName | % | Node CPU as % of allocatable CPU |
| node_memory_utilization | ClusterName, NodeName | % | Node memory as % of allocatable memory |
| node_network_total_bytes | ClusterName, NodeName | B/s | Total network bytes (in + out) per second |
| node_filesystem_utilization | ClusterName, NodeName | % | Node filesystem space used (%) |
| node_number_of_running_pods | ClusterName, NodeName | Count | Running pods on the node |
| pod_cpu_utilization | ClusterName, Namespace, PodName | % | Pod CPU as % of CPU request |
| pod_memory_utilization | ClusterName, Namespace, PodName | % | Pod memory as % of memory request |
| pod_cpu_utilization_over_pod_limit | ClusterName, Namespace, PodName | % | CPU over pod CPU limit (>100% = throttling) |
| pod_memory_utilization_over_pod_limit | ClusterName, Namespace, PodName | % | Memory over pod limit (>100% = OOM kill risk) |
| pod_network_rx_bytes | ClusterName, Namespace, PodName | B/s | Network bytes received per second by pod |
| pod_network_tx_bytes | ClusterName, Namespace, PodName | B/s | Network bytes transmitted per second by pod |
| service_number_of_running_pods | ClusterName, Namespace, Service | Count | Running pods backing a Kubernetes service |
| cluster_node_count | ClusterName | Count | Total nodes in the cluster |
| cluster_failed_node_count | ClusterName | Count | Nodes in failed or not-ready state |
| namespace_number_of_running_pods | ClusterName, Namespace | Count | Running pods in the namespace |

Table H.4 — CloudWatch Container Insights key metrics for EKS/ECS. Enable via aws eks update-cluster-config or the Container Insights console toggle.

**Note: Container Insights Pricing Note**

Container Insights metrics are billed under CloudWatch custom metrics at the standard tiered rate ($0.30/metric/month for first 10,000). A typical 50-node EKS cluster with 200 pods generates approximately 2,000–3,000 Container Insights metrics, adding $600–$900/month to CloudWatch costs. Use CloudWatch metric streams to export Container Insights data to Amazon OpenSearch or third-party observability platforms at lower cost for large clusters.

CHAPTER Appendix I

# CloudWatch Cost Optimization Reference

CloudWatch costs can grow unexpectedly as log volumes, metric cardinality, and dashboards scale. This appendix consolidates the key cost levers, pricing model details, and CLI commands for controlling CloudWatch spend without sacrificing observability.

## I.1 CloudWatch Pricing Components (April 2026)

Understanding which actions incur cost is the first step to optimization. The table below covers the main billable dimensions for CloudWatch in the US East 1 region, which is the price reference region for AWS.

| Component | Pricing Model | Free Tier | Key Cost Driver |
|---|---|---|---|
| Custom Metrics | $0.30/metric/month (1–10k) | 10 metrics/month | High-cardinality dimensions (e.g., per-request IDs) |
| API Calls | $0.01 per 1,000 GetMetricData | 1M API calls/month | Frequent polling from dashboards or scripts |
| Alarms | $0.10/alarm/month (standard) | 10 alarms/month | Too many granular per-resource alarms |
| Dashboard | $3.00/dashboard/month | 3 dashboards/month | Large numbers of rarely viewed dashboards |
| Log Ingestion | $0.50/GB ingested | 5 GB/month | Verbose DEBUG log levels in production |
| Log Storage | $0.03/GB/month | 5 GB/month | Missing or excessively long retention policies |
| Logs Insights | $0.005/GB scanned | None for queries | Full-scan queries over large log groups |
| Contributor Insights | $0.50/rule/month | None | Rules over high-volume log groups |
| Synthetics | $0.0012/canary run | None | High-frequency canaries across many regions |
| Container Insights | $0.035/node/hour (ECS/EKS) | None | Large EKS clusters with per-pod metrics |

Table I.1 — CloudWatch pricing components (us-east-1, April 2026). Verify current prices at aws.amazon.com/cloudwatch/pricing.

## I.2 Identifying Expensive Log Groups

Log ingestion is typically the largest CloudWatch cost driver. The following Logs Insights query identifies the top contributors by data scanned. Run it with a 30-day window to get a representative picture.

```
# Find top 20 log groups by ingested data (last 30 days)
aws logs describe-log-groups \
--query 'logGroups[*].{Name:logGroupName,StoredGB:to_number(storedBytes)/1073741824,Retenti
onDays:retentionInDays}' \
--output table | sort -k3 -rn | head -20

# Find log groups with NO retention policy set (unlimited storage!)
aws logs describe-log-groups \
--query 'logGroups[?!retentionInDays].logGroupName' \
--output text

# Set 90-day retention on ALL log groups missing a policy (bash)
aws logs describe-log-groups \
--query 'logGroups[?!retentionInDays].logGroupName' \
--output text | tr '\t' '\n' | while read lg; do
echo "Setting retention on: $lg"
aws logs put-retention-policy \
--log-group-name "$lg" \
--retention-in-days 90
done

# Check CloudWatch Logs costs in Cost Explorer (last month)
aws ce get-cost-and-usage \
--time-period Start=$(date -d 'first day of last month' +%Y-%m-01),End=$(date +%Y-%m-01) \
--granularity MONTHLY \
--filter '{"Dimensions":{"Key":"SERVICE","Values":["AmazonCloudWatch"]}}' \
--group-by Type=DIMENSION,Key=USAGE_TYPE \
--metrics BlendedCost \
--query 'ResultsByTime[0].Groups[*].{Usage:Keys[0],Cost:Metrics.BlendedCost.Amount}' \
--output table | sort -k2 -rn | head -20
```

## I.3 Reducing Custom Metric Costs

Custom metric costs scale linearly with the number of unique metric name + dimension combinations. The most common optimization mistake is using high-cardinality values (user IDs, request IDs, session tokens) as dimensions.

Warning: High-Cardinality Dimension Anti-Pattern

Avoid using user IDs, transaction IDs, IP addresses, or any unique-per-request values as CloudWatch metric dimensions. Each unique combination creates a new metric time series, which can create millions of metrics and thousands of dollars in monthly charges from a single application. Instead, aggregate at the application level and publish summary metrics (p99, error rate, throughput) with low-cardinality dimensions like Environment, Region, and ServiceName.

```
# List all custom metric namespaces and count metrics per namespace
aws cloudwatch list-metrics \
--query 'Metrics[?Namespace!=`AWS/*`].[Namespace]' \
--output text | sort | uniq -c | sort -rn | head -20

# Count total custom metrics in account
aws cloudwatch list-metrics \
--query 'length(Metrics[?starts_with(Namespace, `AWS`) == `false`])'

# Use Embedded Metrics Format (EMF) to get both logs AND metrics
# (billed as log ingestion only — metrics are free with EMF)
import json
def emit_emf(metric_name, value, unit='Count', namespace='MyApp'):
emf = {
'_aws': {
'Timestamp': int(__import__('time').time() * 1000),

'CloudWatchMetrics': [{
'Namespace': namespace,
'Dimensions': [['Environment', 'Service']],
'Metrics': [{'Name': metric_name, 'Unit': unit}]
}]
},
'Environment': 'production',
'Service': 'checkout-api',
metric_name: value
}
print(json.dumps(emf)) # stdout → CloudWatch Logs → free metrics
```

## I.4 Logs Insights Query Cost Optimization

Logs Insights charges per GB of data scanned. Queries that do not filter by time range, or that run against log groups with no index, scan the maximum possible data. The strategies below reduce scan costs significantly.

| Strategy | Cost Reduction | Implementation |
|---|---|---|
| Always use time range filter | Up to 90% | Use --start-time/--end-time or date picker; avoid all-time queries |
| Filter early with \| filter | 50–80% | Put \| filter before \| stats; reduces data passed to aggregation stage |
| Query only necessary log groups | Variable | Specify exact log group; avoid wildcard /*/ patterns on large groups |
| Use log group index fields | 20–40% | Filter on @logStream, @log which are indexed and scanned cheaply |
| Schedule reports instead of interactive | 60–90% | Use CloudWatch scheduled queries + S3 export instead of ad-hoc |
| Sample high-volume logs | 50–75% | Log 10% of DEBUG events; log 100% of WARN/ERROR events |
| Use metric filters for counting | 95% | Replace count(*) Insights queries with free Metric Filter counters |

Table I.2 — CloudWatch Logs Insights cost optimization strategies.

## I.5 Alarm and Dashboard Cleanup

```
# Find alarms that have been in OK state for 90+ days (candidates for deletion)
aws cloudwatch describe-alarms \
--state-value OK \
--query 'MetricAlarms[?StateUpdatedTimestamp<=`2026-01-15T00:00:00`].[AlarmName,StateUpdatedTimestamp]' \
--output table

# Find alarms with INSUFFICIENT_DATA (metric may not exist anymore)
aws cloudwatch describe-alarms \
--state-value INSUFFICIENT_DATA \
--query 'MetricAlarms[].{Name:AlarmName,Namespace:Namespace,Metric:MetricName}' \
--output table

# Delete an alarm
aws cloudwatch delete-alarms --alarm-names OldAlarmName1 OldAlarmName2

# List all dashboards and their widget counts
aws cloudwatch list-dashboards \
--query 'DashboardEntries[].{Name:DashboardName,Modified:LastModified}' \
--output table

# Delete unused dashboards (30+ days unmodified, example)
```

```
aws cloudwatch list-dashboards \
--query 'DashboardEntries[?LastModified<=`2026-03-01T00:00:00`].DashboardName' \
--output text | tr '\t' '\n' | while read dash; do
echo "Deleting dashboard: $dash"
aws cloudwatch delete-dashboards --dashboard-names "$dash"
done
```

CHAPTER Appendix J

# Terraform Monitoring Baseline Module

This appendix provides a production-ready Terraform module that deploys a complete monitoring baseline for a new AWS account: CloudTrail, Config, GuardDuty, Security Hub, and the foundational CloudWatch alarms. The module is designed to be called from account-vending automation.

## J.1 Module Structure

```
# Directory structure
modules/aws-monitoring-baseline/
■■■ main.tf # Core resources
■■■ variables.tf # Input variables
■■■ outputs.tf # Output values
■■■ versions.tf # Provider version constraints
```

```
# Call the module from account vending root
module "monitoring_baseline" {
source = "./modules/aws-monitoring-baseline"

account_name = "prod-payments"
environment = "production"
log_archive_bucket = "my-org-cloudtrail-logs-111111111111"
alarm_sns_arn = aws_sns_topic.ops_alerts.arn
enable_guardduty = true
enable_security_hub = true
enable_config = true
security_hub_standards = [
"arn:aws:securityhub:us-east-1::standards/aws-foundational-security-best-practices/v/1.0.0"
,
"arn:aws:securityhub:us-east-1::standards/cis-aws-foundations-benchmark/v/1.4.0"
]
tags = {
Owner = "platform-engineering"
CostCenter = "shared-services"
}
}
```

## J.2 variables.tf

```
variable "account_name" { type = string }
variable "environment" { type = string }
variable "log_archive_bucket" { type = string }
variable "alarm_sns_arn" { type = string }
variable "enable_guardduty" { type = bool; default = true }
variable "enable_security_hub" { type = bool; default = true }
variable "enable_config" { type = bool; default = true }
variable "security_hub_standards" {
type = list(string)
default = []
}
variable "cloudtrail_retention_days" { type = number; default = 365 }
variable "log_group_retention_days" { type = number; default = 90 }
variable "tags" { type = map(string); default = {} }
```

## J.3 main.tf — CloudTrail

```
#     ■■     CloudWatch     Log     Group     for     CloudTrail     ■
resource "aws_cloudwatch_log_group" "cloudtrail" {
name = "/aws/cloudtrail/${var.account_name}"
retention_in_days = var.log_group_retention_days
tags = var.tags
}

resource "aws_iam_role" "cloudtrail_cw" {
name = "cloudtrail-cloudwatch-${var.account_name}"
assume_role_policy = jsonencode({
Version = "2012-10-17"
Statement = [{
Action = "sts:AssumeRole"
Effect = "Allow"
Principal = { Service = "cloudtrail.amazonaws.com" }
}]
})
inline_policy {
name = "allow-log-write"
policy = jsonencode({
Version = "2012-10-17"
Statement = [{
```

```
Effect = "Allow"
Action = ["logs:CreateLogStream", "logs:PutLogEvents"]
Resource = "${aws_cloudwatch_log_group.cloudtrail.arn}:*"
}]
})
}
}

# ■■ CloudTrail Trail ■
resource "aws_cloudtrail" "main" {
name = "${var.account_name}-trail"
s3_bucket_name = var.log_archive_bucket
s3_key_prefix = var.account_name
include_global_service_events = true
is_multi_region_trail = true
enable_log_file_validation = true
cloud_watch_logs_group_arn = "${aws_cloudwatch_log_group.cloudtrail.arn}:*"
cloud_watch_logs_role_arn = aws_iam_role.cloudtrail_cw.arn
kms_key_id = aws_kms_key.cloudtrail.arn

event_selector {
read_write_type = "All"
include_management_events = true
data_resource {
type = "AWS::S3::Object"
values = ["arn:aws:s3:::"] # All S3 objects
}
}
tags = var.tags
}

# ■■ KMS Key for CloudTrail encryption ■■
resource "aws_kms_key" "cloudtrail" {
description = "CloudTrail encryption key — ${var.account_name}"
deletion_window_in_days = 30
enable_key_rotation = true
tags = var.tags
policy = data.aws_iam_policy_document.cloudtrail_kms.json
}
```

## J.4 main.tf — GuardDuty, Security Hub, Config

```
# ■■ GuardDuty ■
resource "aws_guardduty_detector" "main" {
count = var.enable_guardduty ? 1 : 0
enable = true
datasources {
s3_logs { enable = true }
kubernetes { audit_logs { enable = true } }
malware_protection { scan_ec2_instance_with_findings { ebs_volumes { enable = true } } }
}
tags = var.tags
}

# ■■ Security Hub ■
resource "aws_securityhub_account" "main" {
count = var.enable_security_hub ? 1 : 0
enable_default_standards = false # Manage standards explicitly
auto_enable_controls = true
}

resource "aws_securityhub_standards_subscription" "standards" {
for_each = toset(var.security_hub_standards)
standards_arn = each.value
depends_on = [aws_securityhub_account.main]
}

# ■■ AWS Config ■
resource "aws_config_configuration_recorder" "main" {
count = var.enable_config ? 1 : 0
name = "default"
role_arn = aws_iam_role.config[0].arn
recording_group {
all_supported = true
include_global_resource_types = true
}
}

resource "aws_config_delivery_channel" "main" {
count = var.enable_config ? 1 : 0
name = "default"
s3_bucket_name = var.log_archive_bucket
```

```
s3_key_prefix = "${var.account_name}/config"
```

```
snapshot_delivery_properties {
delivery_frequency = "Six_Hours"
}
depends_on = [aws_config_configuration_recorder.main]
}

resource "aws_config_configuration_recorder_status" "main" {
count = var.enable_config ? 1 : 0
name = aws_config_configuration_recorder.main[0].name
is_enabled = true
depends_on = [aws_config_delivery_channel.main]
}
```

## J.5 main.tf — Core CloudWatch Alarms

```
# ■■ Root account usage alarm (CIS 3.3) ■
resource "aws_cloudwatch_log_metric_filter" "root_usage" {
name = "root-account-usage"
log_group_name = aws_cloudwatch_log_group.cloudtrail.name
pattern = "{ $.userIdentity.type = \"Root\" && $.userIdentity.invokedBy NOT EXISTS &&
$.eventType != \"AwsServiceEvent\" }"
metric_transformation {
name = "RootAccountUsage"
namespace = "CISBenchmark/${var.account_name}"
value = "1"
}
}

resource "aws_cloudwatch_metric_alarm" "root_usage" {
alarm_name = "${var.account_name}-root-account-usage"
alarm_description = "Root account API call detected — CIS 3.3"
namespace = "CISBenchmark/${var.account_name}"
metric_name = "RootAccountUsage"
period = "300"
evaluation_periods = "1"
threshold = "1"
comparison_operator = "GreaterThanOrEqualToThreshold"
statistic = "Sum"
alarm_actions = [var.alarm_sns_arn]
treat_missing_data = "notBreaching"
tags = var.tags
```

```
}

# ■■ Unauthorized API calls alarm (CIS 3.1) ■
resource "aws_cloudwatch_log_metric_filter" "unauth_api" {
name = "unauthorized-api-calls"
log_group_name = aws_cloudwatch_log_group.cloudtrail.name
pattern = "{ ($.errorCode = \"*UnauthorizedOperation\") || ($.errorCode = \"AccessDenied*\")
}"
metric_transformation {
name = "UnauthorizedAPICalls"
namespace = "CISBenchmark/${var.account_name}"
value = "1"
}
}

resource "aws_cloudwatch_metric_alarm" "unauth_api" {
alarm_name = "${var.account_name}-unauthorized-api-calls"
alarm_description = "Unauthorized API calls detected — CIS 3.1"
namespace = "CISBenchmark/${var.account_name}"
metric_name = "UnauthorizedAPICalls"
period = "300"
evaluation_periods = "1"
threshold = "5"
comparison_operator = "GreaterThanOrEqualToThreshold"
statistic = "Sum"
alarm_actions = [var.alarm_sns_arn]
treat_missing_data = "notBreaching"
tags = var.tags
}
```

## J.6 outputs.tf

```
output "cloudtrail_arn" {
description = "ARN of the CloudTrail trail"
value = aws_cloudtrail.main.arn
}

output "cloudtrail_log_group_name" {
description = "CloudWatch Log Group name for CloudTrail events"
value = aws_cloudwatch_log_group.cloudtrail.name
}

output "guardduty_detector_id" {
```

```
description = "GuardDuty detector ID (empty if disabled)"
value = var.enable_guardduty ? aws_guardduty_detector.main[0].id : ""
}
```

```
output "config_recorder_name" {
description = "AWS Config recorder name (empty if disabled)"
value = var.enable_config ? aws_config_configuration_recorder.main[0].name : ""
}
```

```
output "kms_key_arn" {
description = "ARN of KMS key used for CloudTrail encryption"
value = aws_kms_key.cloudtrail.arn
}
```

## J.7 Deployment and Testing

After writing the module files, the following workflow deploys and validates the baseline in a new AWS account. The deployment typically takes 3–5 minutes for all resources to be created and GuardDuty to begin detecting.

```
# Initialize Terraform (downloads providers)
terraform init
```

```
# Preview changes before applying
terraform plan -var-file=prod-payments.tfvars
```

```
# Apply (requires AdministratorAccess or carefully scoped permissions)
terraform apply -var-file=prod-payments.tfvars -auto-approve
```

```
# Validate CloudTrail is recording
aws cloudtrail get-trail-status \
--name prod-payments-trail \
--query '{IsLogging:IsLogging,LastDelivery:LatestDeliveryTime}'

# Validate GuardDuty is active
DETECTOR=$(aws guardduty list-detectors --query 'DetectorIds[0]' --output text)
aws guardduty get-detector --detector-id $DETECTOR \
--query '{Status:Status,FindingFrequency:FindingPublishingFrequency}'

# Validate Security Hub is active and standards are enabled
aws securityhub get-enabled-standards \
--query 'StandardsSubscriptions[].{Standard:StandardsArn,Status:StandardsStatus}' \
--output table
```

```
# Validate Config recorder is running
aws configservice describe-configuration-recorder-status \
--query 'ConfigurationRecordersStatus[].{Name:name,Recording:recording,Error:lastStatus}'
```

```
# Run GuardDuty sample findings to test alerting pipeline
aws guardduty create-sample-findings \
--detector-id $DETECTOR \
--finding-types \
'UnauthorizedAccess:EC2/SSHBruteForce' \
'Recon:IAMUser/MaliciousIPCaller'

# Verify findings appear in Security Hub
aws securityhub get-findings \
--filters '{"GeneratorId":[{"Value":"guardduty","Comparison":"PREFIX"}]}' \
--query 'Findings[0:5].{Title:Title,Severity:Severity.Label}' \
--output table
```

Note: Module Maintenance

After initial deployment, run terraform plan weekly in CI/CD to detect configuration drift. CloudFormation StackSets and AWS Control Tower's Account Factory for Terraform (AFT) can call this module automatically when new accounts are vended, ensuring every account in the organization starts with the full monitoring baseline from day one.

CHAPTER Appendix K

# AWS Monitoring Operational Runbooks

Operational runbooks provide step-by-step procedures for common monitoring and security operations tasks. These runbooks are designed to be stored in a wiki or runbook automation platform (AWS Systems Manager Documents, Confluence, or Notion) and referenced during incidents and audits.

## K.1 Runbook: New AWS Account Monitoring Checklist

Execute this checklist within 24 hours of any new AWS account creation. The checklist covers the minimum viable monitoring baseline and should be automated via AWS Control Tower or account vending pipeline.

| Step | Action | Validation Command | Owner |
|---|---|---|---|
| 1 | Enable CloudTrail (multi-region, log file validation) | cloudtrail get-trail-status | Platform Eng |
| 2 | Configure CloudTrail → CloudWatch Logs delivery | cloudtrail describe-trails --query '[].CloudWatchLogsLogGroupArn' | Platform Eng |
| 3 | Set CloudWatch Log Group retention (90 days min) | logs describe-log-groups --query '[].retentionInDays' | Platform Eng |
| 4 | Enable GuardDuty and join to org detector | guardduty list-detectors | Security Ops |
| 5 | Enable Security Hub and subscribe to standards | securityhub get-enabled-standards | Security Ops |
| 6 | Enable AWS Config recorder and delivery channel | configservice describe-configuration-recorder-status | Platform Eng |
| 7 | Deploy all 15 CIS benchmark metric filters and alarms | cloudwatch describe-alarms --alarm-name-prefix cis- | Security Ops |
| 8 | Configure SNS alarm topic and verify subscriptions | sns list-subscriptions-by-topic --topic-arn ARN | Platform Eng |
| 9 | Tag all monitoring resources with cost center and owner | resourcegroupstaggingapi get-resources --tag-filters Key=Owner | FinOps |
| 10 | Confirm Security Hub findings flow to SIEM/ticketing | securityhub create-findings (test); verify in SIEM | Security Ops |

| Step | Action | Validation Command | Owner |
|---|---|---|---|
| 11 | Run GuardDuty sample findings test | guardduty create-sample-findings --detector-id ID --finding-types ... | Security Ops |
| 12 | Document account in monitoring inventory | Update monitoring runbook wiki page for account | All |

Table K.1 — New account monitoring checklist. Complete all 12 steps within 24 hours of account creation.

## K.2 Runbook: Responding to a GuardDuty HIGH Finding

This runbook applies to any GuardDuty finding with severity 7.0–8.9 (HIGH). CRITICAL findings (9.0+) require immediate escalation and should follow the incident response runbook (K.3). All actions below should be completed within the SLA for your organization's security tier.

> Note: SLA Targets
>
> HIGH severity GuardDuty findings: Acknowledge within 15 minutes, initial triage within 1 hour, remediation within 4 hours (business hours) or 8 hours (off-hours). CRITICAL findings: immediate 24/7 response — engage on-call engineer immediately.

- 1. Acknowledge: In Security Hub: update finding Workflow Status from NEW to IN_PROGRESS. Assign to yourself. Set note with ticket number.

Command: aws securityhub batch-update-findings --finding-identifiers [...] --workflow Status=IN_PROGRESS --note Text='IR-2026-042' UpdatedBy='analyst'

- 2. Triage: Review full finding detail: affected resource, account, region, finding type. Check if resource is tagged as production/sensitive. Review recent CloudTrail events for the affected resource/principal.

Command: aws guardduty get-findings --detector-id $DETECTOR --finding-ids $FINDING_ID

- 3. Assess false positive: If false positive: mark SUPPRESSED with justification. Create suppression rule to prevent recurrence. Close ticket.

Command: aws securityhub batch-update-findings [...] --workflow Status=SUPPRESSED

- 4. Contain (if real): Isolate affected resource using Appendix G procedures. Disable any IAM principals involved. Notify application team.

Command: See Appendix G for isolation commands

- 5. Investigate: Pull CloudTrail events for affected principal (last 72 hours). Check for lateral movement, data exfiltration, persistence mechanisms.

Command: aws cloudtrail lookup-events --lookup-attributes ...

- 6. Remediate: Remove malware (Inspector), rotate credentials, patch vulnerability, update security groups as appropriate to finding type.

Command: Varies by finding type — see AWS GuardDuty remediation docs

- 7. Close: Update Security Hub finding: Workflow Status = RESOLVED. Document root cause, actions taken, and prevention measures in ticket.

Command: aws securityhub batch-update-findings [...] --workflow Status=RESOLVED

## K.3 Runbook: Monthly Security Review

A consistent monthly review cadence catches configuration drift, validates that monitoring is working, and provides data for security posture reporting to leadership. Allocate 2–4 hours for the monthly review, assign a dedicated DRI (Directly Responsible Individual), and publish findings to the security team wiki.

| Review Area | Commands / Actions | Pass Criteria |
|---|---|---|
| Security Hub score | securityhub get-finding-aggregation-statistics | Score >= 80% on all enabled standards |
| GuardDuty findings trend | Compare findings count by severity vs. prior month | No unresolved HIGH/CRITICAL findings |
| CloudTrail integrity | cloudtrail get-trail-status — check LatestDeliveryError | No delivery errors in trailing 30 days |
| Config compliance | configservice get-compliance-summary-by-config-rule | < 5% non-compliant rules |
| IAM unused credentials | iam generate-credential-report; review last-used dates | No credentials unused > 90 days |
| Root account activity | cloudwatch get-metric-statistics for RootAccountUsage metric | Zero root API calls in review period |
| Log group coverage | logs describe-log-groups \| check for missing retentionInDays | All log groups have retention policy set |
| Alarm health | cloudwatch describe-alarms --state-value INSUFFICIENT_DATA | Zero INSUFFICIENT_DATA alarms |
| Cost review | AWS Cost Explorer: SERVICE = AmazonCloudWatch | Within 10% of prior month budget |
| Access key age | Credential report: check access_key_1/2_last_rotated date | All active keys rotated within 90 days |

Table K.2 — Monthly security monitoring review checklist with pass/fail criteria.

## K.4 Runbook: CloudWatch Alert Tuning

Alert fatigue is one of the biggest threats to an effective security operations program. When security engineers begin ignoring alerts because too many are noisy or false-positive, real incidents go undetected. This runbook covers the process for tuning alerts to achieve a high signal-to-noise ratio.

- Step 1 — Measure noise: For each alarm, track the ratio of actionable to non-actionable triggers over a 30-day period. An alarm triggering more than 5 times per week with no remediation action taken is a candidate for tuning.
- Step 2 — Identify root cause of noise: Common causes include threshold set too low, evaluation period too short, missing treat_missing_data configuration, or the underlying behavior is expected (e.g., deployment-triggered errors).
- Step 3 — Apply remediation: Raise the threshold (e.g., from 1 error to 10 errors in 5 minutes), increase the evaluation period (e.g., 3 of 5 periods), add a maintenance window suppression, or convert to a composite alarm that only fires when multiple conditions are true simultaneously.
- Step 4 — Validate the tuning: After changing alarm parameters, monitor for 2 weeks to confirm the alarm still triggers on real events (use aws cloudwatch set-alarm-state to test) while reducing false positives.
- Step 5 — Document all changes: Record every alarm parameter change in a changelog with the date, reason, previous threshold, and new threshold. This is essential for audit evidence and for reverting changes that over-suppress.

## K.5 Runbook: Quarterly IAM Access Review

A quarterly IAM access review is required by SOC 2 CC6.2 and PCI DSS 7.2. The review identifies stale, overprivileged, and orphaned IAM entities that represent security risk. Block at least half a day to complete the full review for a medium-sized AWS organization.

- Generate Credential Report

```
aws iam generate-credential-report; wait ~30 seconds; aws iam get-credential-report --query
Content --output text | base64 -d > credential-report.csv
```

Review all IAM users. Flag any with password_last_used > 90 days, access key age > 90 days, or MFA disabled.

- Review IAM Access Analyzer

```
aws accessanalyzer list-findings --analyzer-name <arn> --filter
'{"status":{"eq":["ACTIVE"]}}'
```

Resolve all active findings. Each finding represents external access to a resource that may be unintended.

- List unused roles (>90 days)

```
aws iam list-roles --query
'Roles[?RoleLastUsed.LastUsedDate<=`2026-01-15`].[RoleName,RoleLastUsed.LastUsedDate]'
```

Roles unused >90 days are candidates for deletion or deactivation. Verify with application team first.

- Check for admin policies

```
aws iam list-policies --scope Local --query 'Policies[?AttachmentCount>`0`]' | xargs -I{}
aws iam get-policy-version --policy-arn {} --version-id v1
```

Identify any customer-managed policies granting AdministratorAccess or Action:* Resource:*.

- Review service account keys

```
aws iam list-users | xargs -I{} aws iam list-access-keys --user-name {}
```

All programmatic credentials should be rotated. Keys > 90 days old must be rotated immediately.

- Update access review record

```
Update IAM Access Review spreadsheet in security wiki with date, reviewer, and findings
count
```

Formal record required for SOC 2 and PCI DSS audit evidence. Include any exceptions with business justification.

> Note: Automation Opportunity
>
> The quarterly IAM access review can be substantially automated using AWS IAM Access Analyzer, AWS Security Hub IAM controls, and custom Lambda functions that read the credential report and generate a review report automatically. Some organizations automate the access revocation step as well — automatically disabling credentials unused for more than 180 days after sending a warning email at the 90-day mark.

CHAPTER Appendix L

# Security Hub Finding Reference & Remediation

This appendix provides a reference guide for the most commonly triggered Security Hub findings across the AWS Foundational Security Best Practices (FSBP) and CIS AWS Foundations Benchmark standards, with remediation guidance and automation patterns for each.

## L.1 Most Common FSBP Findings and Remediations

The following table lists the top 20 Security Hub FSBP findings by frequency of occurrence across AWS environments, along with automated or CLI-based remediation steps. Findings marked with (A) can be auto-remediated using Security Hub automations or AWS Config auto-remediation.

| Control ID | Finding Title | Severity | Remediation |
|---|---|---|---|
| IAM.1 | IAM policies must not allow full '*' admin | HIGH | Delete or restrict. Use IAM Access Analyzer to generate least-privilege replacement. |
| IAM.3 | Access keys should be rotated every 90 days (A) | MEDIUM | Disable old key with aws iam update-access-key --status Inactive. Create new key. |
| IAM.4 | IAM root user access key must not exist | CRITICAL | Delete root access keys in IAM console. Never use root for API access. |
| IAM.5 | MFA for all IAM users with console password (A) | MEDIUM | Enforce MFA via IAM policy condition: aws:MultiFactorAuthPresent=true. |
| IAM.6 | Hardware MFA for root account | CRITICAL | Enable virtual or hardware MFA for root in IAM Security Credentials console. |
| IAM.21 | Customer policies must not use wildcard actions | HIGH | Use IAM Access Analyzer policy generation to create a least-privilege replacement. |
| S3.1 | Buckets must have Block Public Access enabled (A) | HIGH | aws s3api put-public-access-block --bucket NAME --public-access-block-configuration BlockPublicAcls=true |
| S3.2 | S3 buckets must prohibit public read access (A) | CRITICAL | Remove ACL public-read grants. Enable Block Public Access for the bucket. |
| S3.5 | S3 requests must use SSL (A) | MEDIUM | Add bucket policy Deny where aws:SecureTransport=false. |
| S3.9 | S3 server access logging must be enabled | MEDIUM | aws s3api put-bucket-logging --bucket NAME --logging-config TargetBucket=LOG-BUCKET |

| Control ID | Finding Title | Severity | Remediation |
|---|---|---|---|
| EC2.1 | EBS snapshots not publicly restorable | CRITICAL | aws ec2 disable-image-block-public-access; revoke createVolumePermission on snapshots. |
| EC2.2 | VPC default SG must not allow traffic (A) | HIGH | Remove all rules from default security group. Use named SGs per workload. |
| EC2.6 | VPC flow logging must be enabled (A) | MEDIUM | aws ec2 create-flow-logs --resource-type VPC --resource-ids VPC-ID --traffic-type ALL |
| EC2.7 | EBS default encryption must be enabled (A) | MEDIUM | aws ec2 enable-ebs-encryption-by-default --region REGION |
| EC2.15 | Subnets must not auto-assign public IPs (A) | MEDIUM | aws ec2 modify-subnet-attribute --subnet-id ID --no-map-public-ip-on-launch |
| RDS.2 | RDS instances must not be publicly accessible (A) | HIGH | aws rds modify-db-instance --db-instance-identifier ID --no-publicly-accessible |
| RDS.3 | RDS storage encryption must be enabled | MEDIUM | Encryption is set at creation only. Restore to a new encrypted instance if needed. |
| CloudTrail.1 | CloudTrail: one multi-region trail required | HIGH | aws cloudtrail create-trail --is-multi-region-trail --include-global-service-events |
| CloudTrail.2 | CloudTrail encryption at rest required | MEDIUM | aws cloudtrail update-trail --name TRAIL --kms-key-id KEY-ARN |
| GuardDuty.1 | GuardDuty must be enabled | HIGH | aws guardduty create-detector --enable; configure org delegated admin. |

Table L.1 — Top Security Hub FSBP findings by frequency, with remediation guidance. (A) = auto-remediable via Security Hub automations.

## L.2 Security Hub Automation Rules (Auto-Remediation)

Security Hub Automation Rules (GA since 2023, continuously expanded in 2025–2026) allow findings to be automatically updated or suppressed based on criteria. The following Terraform snippet creates an automation rule that auto-suppresses S3 block public access findings for known-public buckets tagged with PublicBucket=true.

```
# Terraform: Security Hub Automation Rule
resource "aws_securityhub_automation_rule" "suppress_public_buckets" {
rule_name = "suppress-intentional-public-s3"
rule_order = 100
description = "Suppress S3.1/S3.2 for buckets tagged PublicBucket=true"
is_terminal = false # Continue evaluating other rules

criteria {
```

```
generator_id {
comparison = "PREFIX"
value = "aws-foundational-security-best-practices/v/1.0.0/S3"
}
resource_tags_key { comparison = "EQUALS"; value = "PublicBucket" }
resource_tags_value { comparison = "EQUALS"; value = "true" }
}

actions {
type = "FINDING_FIELDS_UPDATE"
finding_fields_update {
workflow { status = "SUPPRESSED" }
note {
text = "Suppressed: intentionally public bucket — tagged PublicBucket=true"
updated_by = "SecurityHub-AutoRule"
}
}
}
}
```

## L.3 ASFF (Amazon Security Finding Format) Field Reference

All Security Hub findings conform to ASFF (Amazon Security Finding Format). Understanding key ASFF fields is essential for writing custom integrations, SIEM rules, and automation logic.

| ASFF Field | Type | Description / Common Values |
|---|---|---|
| SchemaVersion | String | Always '2018-10-08' |
| Id | String | Unique finding ARN: arn:aws:securityhub:region:acct:finding/id |
| ProductArn | String | ARN of product generating finding (e.g., .../aws/guardduty) |
| GeneratorId | String | Rule/check ID (e.g., cis-aws-foundations-benchmark/v/1.4.0/1.1) |
| AwsAccountId | String | 12-digit account ID where finding was generated |
| Types | String[] | Classification (e.g., Software and Configuration Checks/CIS) |
| Severity.Label | Enum | INFORMATIONAL, LOW, MEDIUM, HIGH, CRITICAL |
| Severity.Normalized | Integer | 0–100 numeric severity score |
| Title | String | Human-readable finding title (max 256 chars) |
| Description | String | Detailed finding description (max 1024 chars) |

| ASFF Field | Type | Description / Common Values |
|---|---|---|
| Remediation.Recommendation.Text | String | Short remediation guidance text |
| Remediation.Recommendation.Url | String | URL link to remediation documentation |
| Resources[].Type | String | ASFF type (e.g., AwsS3Bucket, AwsEc2Instance) |
| Resources[].Id | String | ARN of the affected resource |
| Compliance.Status | Enum | PASSED, FAILED, NOT_AVAILABLE, WARNING |
| Workflow.Status | Enum | NEW, NOTIFIED, IN_PROGRESS, RESOLVED, SUPPRESSED |
| RecordState | Enum | ACTIVE or ARCHIVED |
| CreatedAt | ISO8601 | Timestamp when finding was first created |
| UpdatedAt | ISO8601 | Timestamp when finding was last updated |
| FirstObservedAt | ISO8601 | When issue first observed (may predate CreatedAt) |

Table L.2 — Key ASFF fields for Security Hub finding integration and automation.

## L.4 Querying Security Hub Findings with the CLI

```
# Get all CRITICAL findings in the last 7 days
aws securityhub get-findings \
--filters '{
"SeverityLabel": [{"Value":"CRITICAL","Comparison":"EQUALS"}],
"RecordState": [{"Value":"ACTIVE","Comparison":"EQUALS"}],
"WorkflowStatus":[{"Value":"NEW","Comparison":"EQUALS"}],
"CreatedAt": [{"DateRange":{"Value":7,"Unit":"DAYS"}}]
}' \
--query 'Findings[].{Title:Title,Account:AwsAccountId,Resource:Resources[0].Id}' \
--output table

# Count findings by severity across all accounts (aggregator)
aws securityhub get-findings \
--filters '{"RecordState":[{"Value":"ACTIVE","Comparison":"EQUALS"}]}' \
--query 'Findings[].Severity.Label' \
--output text | tr '\t' '\n' | sort | uniq -c

# Export findings to JSON for SIEM ingestion
aws securityhub get-findings \
--filters '{"RecordState":[{"Value":"ACTIVE","Comparison":"EQUALS"}]}' \
```

```
--output json > securityhub-findings-$(date +%Y%m%d).json

# Get Security Hub consolidated security score
aws securityhub get-insight-results \
--insight-arn arn:aws:securityhub:::insight/securityhub/default/35 \
--query 'InsightResults.ResultValues[0:5]'
```