MODULE 9 · STUDY NOTES

Monitoring and Security

How Amazon CloudWatch, AWS CloudTrail, AWS Config, and Amazon Athena monitor performance, audit activity, and keep AWS resources secure

Operating in the cloud means you cannot walk over to a server and check on it — you keep infrastructure healthy and secure through observability and auditing. This module assembles the AWS toolkit for both. **Amazon CloudWatch** is the monitoring hub, pulling metrics, logs, and events from more than 70 services into one operational view, raising alarms, and reacting to change in near real time. **AWS CloudTrail** answers the security question CloudWatch cannot — who did what, and when — by auditing the API calls behind every Console and CLI action and archiving them to Amazon S3. **AWS Config** tracks whether your resources stay configured the way your policies require, and **Amazon Athena** lets you query all of those logs sitting in S3 with plain SQL. Monitoring and security are two sides of the same discipline here: the same services that tell you a system is slow also tell you when someone is attacking it, and the recurring best practice is to use CloudWatch and CloudTrail together.

LEARNING OBJECTIVES

- Explain the benefits of Amazon CloudWatch
- Describe CloudWatch monitoring features, including metrics and alarm details
- Describe Amazon CloudWatch Logs features and benefits
- Explain the purpose and function of AWS CloudTrail
- Describe AWS Config features and benefits
- Use Amazon CloudWatch to monitor applications and infrastructure

9.1 Amazon CloudWatch

Amazon CloudWatch is a monitoring and management service that provides you with data and actionable insights to monitor your applications, understand and respond to system-wide performance changes, optimize resource utilization, and get a unified view of operational health. It works by collecting monitoring and operational data in three forms — **logs, metrics, and events** — from the AWS resources, applications, and services that run on AWS. From that single vantage point you can set alarms, visualize logs and metrics side by side, take automated actions, troubleshoot issues, and discover ways to keep applications running smoothly. CloudWatch is integrated with **more than 70 AWS services** — such as Amazon EC2, Amazon S3, Amazon ECS, AWS Lambda, and Amazon API Gateway — that automatically publish detailed standard metrics with no action on your part.

THE THREE FUNCTIONAL AREAS OF CLOUDWATCH

- **CloudWatch monitoring** — collects metrics on the state and utilization of your resources, and raises alarms and notifications when thresholds are crossed.
- **CloudWatch Logs** — collects, stores, and monitors log files from EC2 instances, CloudTrail, Route 53, and other sources, in near real time.
- **CloudWatch Events** — delivers a near-real-time stream of system events describing changes in AWS resources, and routes them to targets for automated responses.

COMMON PITFALL

CloudWatch is more than EC2 CPU graphs. Its value is the unified view — logs, metrics, and events together — across the whole AWS ecosystem, which is what lets you both optimize performance and spot security problems from one service.

9.2 CloudWatch monitoring: metrics and alarms

CloudWatch monitoring watches the **state and utilization of most of the resources you manage under AWS**. Its key concepts are **standard metrics, custom metrics, alarms, and notifications**, and its data comes either automatically from AWS services or from the **CloudWatch agent**, which collects system-level metrics from EC2 instances and even from on-premises servers. **Standard metrics** are published automatically: EC2 instances emit CPU utilization, data transfer, and disk usage; managed Amazon RDS databases expose memory and disk activity; AWS Lambda offers eight built-in metrics to detect errors and throttles. **Custom metrics** cover what AWS cannot see for you — for example, the memory usage of an `httpd` service running on an EC2 instance, or user activity tracked over time — which you publish yourself through the AWS CLI, the API, or the CloudWatch agent.

KEY TERMS

Metric
A time-ordered set of data points published to CloudWatch — think of it as a variable to monitor, whose data points are its values over time (e.g. one instance's CPU utilization).

Namespace
A container that isolates metrics so metrics from different applications are never aggregated together. AWS namespaces follow the convention `AWS/<service>` — for example, Amazon EC2 uses `AWS/EC2`.

Dimension
A name-value pair that uniquely identifies a metric. You can assign up to 10 dimensions to a metric and use them to filter results — for example, get statistics for one instance with the `InstanceId` dimension.

Data point
A single value carrying a time stamp and, optionally, a unit of measure.

Period
The length of time associated with a statistic, expressed in seconds. A period can be as short as 1 second or as long as 1 day (86,400 seconds); varying it changes how the data is aggregated.

Two properties of metrics catch people out. First, **metrics exist only in the Region where they are created**. Second, **metrics cannot be deleted** — they simply **expire after 15 months** if no new data is published to them, with data points older than 15 months dropping off on a rolling basis as new ones arrive.

TABLE 9.1 Standard metrics vs. custom metrics

ASPECT	STANDARD METRICS	CUSTOM METRICS
Source	Published automatically by 70+ AWS services	Your own data — you publish them
Grouped by	Service name (e.g. all Amazon EC2 metrics together)	A user-defined namespace
How published	Automatically; no action required	AWS CLI, API, or the CloudWatch agent
Availability	Only if the service was used in the past 15 months	As soon as you publish data
Cost	Free	Paid

> **WORKED EXAMPLE – A MONITORING SETUP IN ACTION**
>
> An EC2 instance has the CloudWatch agent installed and detailed monitoring enabled. It sends two metrics: **CPU utilization**, a standard metric collected easily, and **memory utilization of the httpd service**, which is not visible at the hypervisor layer and so must be defined as a **custom metric**. A CloudWatch **alarm** is configured to trigger whenever CPU utilization exceeds a set percentage. When it fires, the alarm publishes to an **Amazon SNS** topic that generates an email notification and reaches a third-party paging system, which pages on-shift IT staff; the same alarm also drops a message on an **Amazon SQS** queue that creates a work item. One condition, several coordinated responses.

Alarms are how monitoring becomes action. A CloudWatch alarm watches a single metric — or the result of a **math expression** based on multiple metrics — and performs one or more actions based on the value relative to a threshold over a number of time periods, testing greater-than-or-equal-to or less-than-or-equal-to comparisons. Critically, the **ALARM state is not necessarily an emergency**: it simply signals that the monitored metric is above, below, or equal to a threshold. You might, for example, alarm when a T2 instance's `CPUCreditBalance` is running low, then process that notification programmatically to suspend a CPU-intensive job until the credit balance recovers.

TABLE 9.2 The three alarm states

STATE	WHAT IT MEANS
OK	The metric is within the defined threshold
ALARM	The metric is outside the defined threshold — not necessarily an emergency
INSUFFICIENT_DATA	The alarm just started, the metric is not available, or there is not enough data to determine the state

An alarm can also require a threshold to be breached for several **consecutive periods** before it acts, which filters out momentary spikes. If the threshold is 3 and the minimum breach is 3 periods, a value that stays above 3 for the third through fifth periods sets the state to ALARM; a single dip below returns it to OK; and a lone breach later — without three consecutive periods — leaves the state at OK.

Beyond individual metrics and alarms, **CloudWatch dashboards** are customizable home pages in the console that surface data about your running AWS ecosystem in a single view and can be leveraged by existing monitoring tools. **Automatic dashboards** are pre-built with recommended best practices for AWS services; they stay resource-aware, update dynamically to reflect the latest state, and let you drill from an account- or resource-level view down to the root cause of a performance issue. By default, EC2 instances report data every 5 minutes as part of the **basic monitoring**

included with the AWS Free Tier; enabling **detailed monitoring** on an instance raises the reporting frequency to once every minute, for an extra charge.

USING CLOUDWATCH FOR SECURITY

- Monitor for **suspicious activity** — unusual, prolonged spikes in service usage such as CPU, disk activity, or Amazon RDS usage.
- Set **alarms on billing metrics** to catch compromised credentials early — some customers only discover that their IAM access keys were stolen when a bill for thousands of dollars of unexpected charges arrives.
- Billing alerts must be **enabled in your account settings** before you can alarm on estimated charges for the month.

COMMON PITFALL

Standard metrics are free, but detailed monitoring and custom metrics are paid. Do not assume every CloudWatch feature is included at no cost — basic 5-minute monitoring is complimentary, while 1-minute detailed monitoring and any metric you publish yourself incur charges.

9.3 Amazon CloudWatch Events

Amazon CloudWatch Events delivers a near-real-time stream of system events that describe changes in AWS resources. Using simple rules you configure, it matches incoming events and routes them to one or more target functions or streams — becoming aware of operational changes as they occur and responding by sending messages, activating functions, making changes, or capturing state information. You can also use CloudWatch Events to **schedule automated actions** that self-trigger at set times using cron or rate expressions.

KEY TERMS

Event
An indication of a change in your AWS environment. Resources emit events when their state changes — for example, Amazon EC2 generates an event when an instance moves from `pending` to `running`. You can also publish custom application-level events or schedule periodic events.

Target
A component that processes events. Example targets include Amazon EC2 instances, AWS Lambda functions, Amazon SNS topics, and Amazon SQS queues.

Rule
A matcher that routes incoming events to targets for processing. A single rule can route to multiple targets, all processed in parallel, so different parts of an organization can act on the events they care about.

WORKED EXAMPLE – REACT TO A NEW INSTANCE

A CloudWatch Events rule watches for the creation of a new EC2 instance. Every time one is created, the rule runs an **AWS Systems Manager Run Command** script on the instance — for example, to install an agent or apply a baseline configuration automatically, with no human in the loop.

KEY TAKEAWAY

- Amazon CloudWatch Events can **trigger services** — such as AWS Lambda — based on **near-real-time events**.

9.4 Amazon CloudWatch Logs

Amazon CloudWatch Logs lets you monitor, store, and access log files from Amazon EC2 instances, AWS CloudTrail, Amazon Route 53, and other sources, then retrieve that log data for analysis. You can monitor logs in near real time for specific phrases, values, or patterns — for instance, alarming on the number of errors in your EC2 system logs, or graphing the latency of web requests parsed from application logs. Because log data can be stored and accessed indefinitely and lives **externally to the EC2 instances**, you never lose valuable information when an instance is terminated and you never worry about filling up local disks.

> **THE THREE PHASES OF LOG ANALYSIS**
>
> - **Configure** — decide what information you need to capture in your logs, and where and how it will be stored.
> - **Collect** — because instances come and go in the cloud, adopt a strategy for periodically uploading a server's log files so the data survives instance termination.
> - **Analyze** — turn collected data into visibility into daily system health, upcoming trends in customer behavior, and insight into how customers use your system.

Functionally, CloudWatch Logs **automatically collects logs** from supported sources — you install the unified CloudWatch agent (or the older CloudWatch Logs agent) on any EC2 instance you want to collect from — and **aggregates data into log groups**, where each log group represents a specific type of log with a set format. On a log group you can configure a **metric filter** that looks for specific string patterns; each match is assigned a numeric value that **increments a custom CloudWatch metric**, which you then use exactly like any other custom metric to create alarms or send notifications. For deeper investigation, **CloudWatch Logs Insights** provides a purpose-built query language with sample queries, autocompletion, and log-field discovery.

> **WORKED EXAMPLE – ALARMING ON 404 ERRORS**
>
> A log group named `HttpAccessLog` holds aggregated data collected by the CloudWatch agents on one or more EC2 instances. An administrator creates a custom metric filter on the group to search for `404` page-not-found errors. Each match increments a custom metric, `404Count`. The administrator then triggers a CloudWatch alarm whenever `404Count` exceeds a specified value, and the alarm sends a notification — here, to the IT support team. Log content becomes a measurable, alertable metric.

To parse logs you must understand their **format**. An Apache web server writes an access log (typically `/var/log/httpd/access_log`) configured by a substitution string such as `%h %l %u %t "%r" %>s %b`, producing a space-delimited line per request — for example `127.0.0.1 - frank [10/Oct/2000:13:55:36 -0700] "GET /apache_pb.gif HTTP/1.0" 200 2326`. Here `%h` is the client IP, `%u` the user ID, `%t` the time, `%r` the request line, `%>s` the HTTP status code (200 success, 404 not found), and `%b` the size of the object returned.

Filter patterns are how you pull relevant data out of those lines, and two rules govern them. They are **case-sensitive**, and while **multiple terms are allowed**, **all terms must appear in a single log event** for it to match — a pattern of `ERROR Exception` matches only entries that contain both words. You create a metric filter by selecting a log group in the CloudWatch console and choosing Create Metric Filter, then defining the pattern that complies with the filter syntax.

> **KEY TAKEAWAY**
>
> - Amazon CloudWatch Logs can **collect application logs** and turn them into monitoring signals by **filtering metric data points for events**.

9.5 AWS CloudTrail

AWS CloudTrail generates logs of calls to the AWS API. Because the AWS API underlies both the AWS CLI and the AWS Management Console, CloudTrail can record essentially **all activity against the services it monitors** — enabling governance, compliance, operational auditing, and risk auditing of your AWS accounts. It logs, continuously monitors, and retains account activity related to actions across your infrastructure; it is supported for a large and growing number of AWS services; and once configured, it **automatically pushes its auditing logs to Amazon S3**. One important boundary: CloudTrail does **not track events that occur within an EC2 instance** — it will not record when someone opens an SSH session and issues `sudo shutdown -h now`.

QUESTIONS A CLOUDTRAIL TRAIL CAN ANSWER

- Why was a long-running instance terminated, and who terminated it? (organizational traceability and accountability)
- Who changed a security group configuration? (accountability and security auditing)
- Is any activity coming from an unknown IP address range? (possible external attack)
- What activities were denied due to lack of permissions? (possible internal or external attack attempts)

By default, CloudTrail **event history** shows only the results from the **last 90 days**, limited to management events (create, modify, and delete API calls) and account activity. For a complete record — all management events, data events, and read-only activity — you must **configure a trail**, which you can create through the CloudTrail console or the AWS CLI.

STEPS TO CONFIGURE A TRAIL

- Configure a **new or existing Amazon S3 bucket** to receive the log files.
- Define the trail and choose which events to log — read-only, write-only, or all; **all management events are logged by default**.
- Create an **Amazon SNS topic** to receive notifications when log files are delivered.
- Optionally, send the logs to **CloudWatch Logs** so you can monitor for specific log events.
- Optionally, turn on **log file encryption and integrity validation**.
- Optionally, add **tags** (custom key-value pairs) to the trail.

WORKED EXAMPLE – READING A CLOUDTRAIL LOG ENTRY

A CloudTrail log file is JSON and can contain one or more entries; each entry represents a single request plus its response, and entries are **not guaranteed to be in order** (they are not an ordered stack trace). The `requestParameters` portion identifies who and what: `userIdentity` (who or what took the action), `eventTime` (when), `eventSource` (whether it came from the console, CLI, or an API call), `eventName`, `awsRegion`, and `sourceIPAddress`. The `responseElements` portion records the outcome — for a console sign-in, `ConsoleLogin` reads `Success` or `Failure`, while `additionalEventData` notes details such as whether MFA was used.

For monitoring and security, **CloudWatch and CloudTrail are complementary, and using both is a best practice.** You can examine the entries from CloudWatch Logs and CloudTrail together to detect potential unauthorized use — for example, failed AWS Management Console sign-in attempts (especially from suspicious IP addresses), unauthorized access to services via API calls, and suspicious launches of AWS resources.

COMMON PITFALL

CloudTrail audits the API, not the operating system. It records who called which AWS API and from where, but anything that happens strictly inside the instance — a manual shutdown, a local process, an edited file — is invisible to it. For in-instance visibility you need OS-level logging shipped through the CloudWatch agent.

9.6 AWS service integration with Amazon Athena

Amazon Athena is an interactive query service that makes it easy to analyze data in Amazon S3 using standard SQL. You point Athena at your data in S3, define the schema, and start querying — most results return within seconds. Athena is **serverless**, so there is no infrastructure to manage and **no need for complex ETL** jobs to prepare data for analysis; you **pay only for the queries you run**; and because it automatically runs queries in parallel, large-scale datasets come back quickly. It works with standard formats including CSV, JSON, ORC, Avro, and Parquet, and it uses Amazon S3 as its durable, highly available underlying store.

GETTING STARTED WITH AMAZON ATHENA

- Create an Amazon S3 bucket and load data into it — or use an existing bucket that already holds data.
- Define the **schema** (a table definition describing the data structure); the definition ends with a `LOCATION` statement pointing to the S3 bucket.
- Start querying with standard SQL in the built-in query editor; results display there and can also be downloaded as CSV.

Athena earns its place in a monitoring-and-security module because it **integrates with other AWS services to make their logs easy to query**. That lets you turn raw log archives in S3 into investigative answers.

TABLE 9.3 AWS logs you can query with Athena

LOG SOURCE	WHAT QUERYING IT REVEALS
AWS CloudTrail logs	Identify trends and isolate account activity by attribute, such as source IP address or user; tables can be created directly from the CloudTrail console
Application Load Balancer logs	See the source of traffic, latency, and bytes transferred to and from load balancers and backend applications
Amazon VPC Flow Logs	Investigate network traffic patterns and identify threats and risks across the Amazon VPC network

9.7 AWS Config

It is not uncommon for a customer to run hundreds — even thousands — of EC2 instances, and other resource types may exist in similarly large numbers. Managing the configuration of all of them presents a real challenge: you may know what the current configuration *should* be, but do you have **visibility into possible configuration errors**, and a system for **managing and tracking configuration changes**? That is the business problem **AWS Config** solves.

AWS Config is a service that enables you to assess, audit, and evaluate the configurations of your AWS resources. It **continuously monitors and records** your resource configurations and lets you **automate the evaluation of recorded configurations against desired configurations**.

WITH AWS CONFIG YOU CAN

- Retrieve an **inventory** of AWS resources.
- Discover **new and deleted** resources.
- **Record configuration changes continuously** and determine your overall compliance against the configurations specified by internal guidelines.
- Get **notified when configurations change**, and dive into detailed resource configuration histories.

Together these features simplify **compliance auditing, security analysis, change management, and operational troubleshooting.** The mechanism is a **rule system**: you can use existing rules from AWS and partners, or define your own **custom rules using AWS Lambda**. Rules can be targeted at specific resources, specific resource types, or resources tagged in a particular way; they run when those resources are **created or changed** and can also be evaluated on a **periodic basis** (hourly, daily, and so on). After setup, AWS Config is invoked automatically so configurations are constantly assessed, and it provides a **dashboard** for visualizing compliance and identifying changes of concern.

EXAMPLE CONFIG RULES

Rules can look for any desirable or undesirable condition. For example, you could define rules that ensure Amazon EBS volumes are **encrypted**, that instances are launched only from **approved AMIs**, that Elastic IP addresses are **attached to instances**, and that EC2 instances are **properly tagged**.

COMMON PITFALL

Config is about configuration, not performance. When a question asks whether resources match a required setup, track configuration changes over time, or prove compliance, the answer is AWS Config — CloudWatch handles metrics and performance, and CloudTrail handles the record of who made the API call.

Module summary

- Amazon CloudWatch collects logs, metrics, and events from 70+ AWS services into a unified operational view; its three areas are CloudWatch monitoring (metrics + alarms), CloudWatch Logs, and CloudWatch Events.
- A metric is defined by a name, a namespace (`AWS/<service>`), and up to 10 dimensions; data points carry a time stamp and optional unit; a period runs 1 second to 1 day. Metrics are Regional, cannot be deleted, and expire after 15 months.
- Standard metrics publish automatically and are free; custom metrics (via CLI, API, or the CloudWatch agent) and detailed monitoring cost extra. Basic monitoring reports every 5 minutes, detailed monitoring every 1 minute.
- Alarms test a metric or math expression against a threshold over periods and hold one of three states — OK, ALARM, INSUFFICIENT_DATA; ALARM is not an emergency. They notify via SNS/SQS or trigger automated actions; enable billing alerts to catch compromised credentials.
- CloudWatch Events streams near-real-time resource changes; rules match events and route them in parallel to targets (EC2, Lambda, SNS, SQS), and can be scheduled with cron or rate expressions.
- CloudWatch Logs collects logs (via the agent), aggregates them into log groups, applies metric filters that increment custom metrics for alarms, and queries with Logs Insights; filter patterns are case-sensitive and require every term to appear in a log event.

- AWS CloudTrail audits API calls (Console, CLI, SDK) and pushes logs to Amazon S3; event history covers the last 90 days, so create a trail for a full record. It does not record actions inside an EC2 instance.
- CloudWatch and CloudTrail are complementary — use both to detect unauthorized use (failed sign-ins, suspicious IPs, unauthorized API calls, suspicious launches).
- Amazon Athena runs serverless standard-SQL queries over data in S3 — including CloudTrail, Application Load Balancer, and VPC Flow Logs — to investigate activity and threats.
- AWS Config assesses, audits, and evaluates resource configurations, continuously recording changes and checking compliance with managed or custom (Lambda) rules that run on change or periodically, all surfaced on a compliance dashboard.

Review questions

1. **What three kinds of data does CloudWatch collect, and what are its three functional areas?**

   Answer: Data: logs, metrics, and events. Areas: CloudWatch monitoring (metrics and alarms), CloudWatch Logs, and CloudWatch Events.

2. **What uniquely defines a metric, and what are a namespace, a dimension, and a period?**

   Answer: A name, a namespace, and up to 10 dimensions. A namespace isolates metrics (AWS/<service>); a dimension is a name-value pair that filters a metric; a period is a statistic's time span, 1 second to 1 day (86,400 s).

3. **How do standard and custom metrics differ in source, grouping, and cost?**

   Answer: Standard: published automatically by AWS, grouped by service, free, and shown only if the service was used in the past 15 months. Custom: your own data, a user-defined namespace, published via CLI/API/agent, and paid.

4. **What are the three alarm states, and does ALARM mean there is an emergency?**

   Answer: OK, ALARM, and INSUFFICIENT_DATA. No — ALARM only means the monitored metric crossed the threshold; it can be a normal, expected condition you handle programmatically.

5. **Walk a log line through the CloudWatch Logs pipeline to an alert.**

   Answer: The agent ships logs into a log group; a metric filter matches a string pattern and increments a custom metric on each match; a CloudWatch alarm on that metric sends an Amazon SNS notification when it exceeds the threshold.

6. **What does CloudTrail record, where does it store logs, and what does it not record?**

   Answer: It records API calls across AWS services (behind the Console, CLI, and SDK) and pushes the logs to Amazon S3. It does not record in-instance actions like a manual SSH shutdown; event history is 90 days, so create a trail for a complete record.

7. **How do CloudWatch and CloudTrail complement each other for security?**

   Answer: CloudWatch Logs monitors your application and system log content in near real time; CloudTrail is the API audit trail of who did what. Using both is the best practice — together they detect failed sign-ins, suspicious IPs, unauthorized API access, and suspicious resource launches.

8. **What does AWS Config do, and how do its rules work?**

   Answer: It assesses, audits, and evaluates AWS resource configurations, continuously recording configurations and changes and checking compliance against desired configurations. Rules (managed or custom Lambda) target resources, types, or tags, run on change or periodically, and report compliance on a dashboard.