AWS CloudWatch

Verified 2026-11-02

Logs, metrics, alarms and Logs Insights for application teams — what to log, what to make a metric, and how not to get a surprise bill.

Contents

Mental model 2

Log groups and retention 2

Structured logs and embedded metric format 3

Logs Insights queries 4

Metrics, namespaces and dimensions 5

Alarms and composite alarms 6

Dashboards and service defaults 7

Cost traps 8

Further reading 9

Mental model

Metrics and logs are two different products with opposite economics. A **metric** is a pre-aggregated time series: cheap to query and alarm on, but every unique combination of dimension values is billed as a separate metric. A **log** is the raw event: cheap to keep, paid for when ingested and again when queried. Design deliberately which few facts become metrics, and leave everything else in logs.

Log groups and retention

Concept	Meaning
Log group	One app or function, e.g. `/aws/lambda/api`
Log stream	One source inside it: instance, task, container
Retention	Never expire by default; 1 day – 10 years
Log class	Standard, or Infrequent Access (cheaper ingest)

Retention is set per log group. Standard and Infrequent Access differ only in ingestion price — storage and Insights pricing are the same — and Infrequent Access supports a subset of features. You cannot change a group's class after creation.

```
aws logs put-retention-policy \
  --log-group-name /aws/lambda/api \
  --retention-in-days 30
aws logs tail /aws/lambda/api \
  --follow --since 10m
```

> **Gotcha:** Log groups that Lambda and other services create for you keep logs forever. Set retention on every group, or storage grows without bound.

Structured logs and embedded metric format

Field	Purpose
`_aws.CloudWatchMetrics`	Declares which fields are metrics
`Namespace`	Your metric namespace
`Dimensions`	Sets of keys, max 30 per set
Root members	Dimension values and metric values

Log one JSON object per line and Logs Insights discovers its fields automatically. **Embedded metric format** (EMF) goes one step further: a JSON log line with an `_aws` block makes CloudWatch extract metrics from it, with no `PutMetricData` call. Extra fields like `requestId` stay searchable in the log.

```
{
  "_aws": {
    "Timestamp": 1574109732004,
    "CloudWatchMetrics": [{
      "Namespace": "shop",
      "Dimensions": [["route"]],
      "Metrics": [{ "Name": "latency",
        "Unit": "Milliseconds" }]
```

```
    }]
  },
  "route": "/checkout",
  "latency": 142,
  "requestId": "989ffbf8-9ace"
}
```

> **Warning:** Every dimension set creates metrics per unique value combination. Put `requestId` or `userId` in `Dimensions` and you create one billed custom metric per request.

Logs Insights queries

Command	Does
`fields`	Choose or compute fields
`filter`	Keep matching events
`stats … by`	Aggregate, e.g. `count(*)`, `avg()`, `bin(5m)`
`sort` / `limit`	Order and cap results
`parse`	Extract fields with a glob or regex
`dedup`	Keep the first event per value

Queries are pipelines of commands joined with `|`. System fields start with `@`: `@timestamp`, `@message`, `@logStream`, and for Lambda `@type`, `@duration` and `@requestId`. Queries time out after 60 minutes, and an account runs at most 100 concurrent Insights queries, dashboard queries included.

```
fields @timestamp, @requestId, @duration
| filter @type = "REPORT" and @duration > 1000
| sort @timestamp desc
| dedup @requestId
| limit 20
```

```
filter level = "error"
| stats count(*) as errors by route, bin(5m)
```

> **Gotcha:** Insights bills by uncompressed data scanned, not rows returned. `limit 20` over a month of logs still scans the month — narrow the time range and log groups first.

Metrics, namespaces and dimensions

Retention	Resolution
3 hours	Under 60 s (high resolution)
15 days	1 minute
63 days	5 minutes
455 days (15 months)	1 hour

A metric is identified by namespace, name and dimensions; AWS services publish to namespaces like `AWS/Lambda` and `AWS/ECS`. Older data is rolled up automatically, so after 15 days you see 5-minute points, not 1-minute ones. Metrics are per Region and cannot be deleted; one that gets no new data expires after 15 months.

```
aws cloudwatch put-metric-data \
  --namespace shop \
  --metric-name orders \
  --dimensions route=/checkout \
  --value 1 --unit Count
```

Alarms and composite alarms

Setting	Meaning
States	`OK`, `ALARM`, `INSUFFICIENT_DATA`
M out of N	Datapoints to alarm of evaluation periods
`TreatMissingData`	`missing` (default), `ignore`, `breaching`, `notBreaching`
Composite rule	`ALARM("a") AND NOT ALARM("b")`

An alarm watches one metric or metric-math expression and fires actions — SNS, Lambda, autoscaling — on state change. Use M out of N, such as 3 of 5, to ride out single spikes. A **composite alarm** combines other alarms with `AND`, `OR` and `NOT`, so you page once for “API is down” instead of five times.

```
aws cloudwatch put-metric-alarm \
  --alarm-name api-errors \
  --namespace AWS/Lambda --metric-name Errors \
  --dimensions Name=FunctionName,Value=api \
  --statistic Sum --period 60 \
```

```
  --evaluation-periods 5 --datapoints-to-alarm 3 \
  --threshold 5 \
  --comparison-operator GreaterThanThreshold \
  --treat-missing-data notBreaching \
  --alarm-actions arn:aws:sns:eu-west-1:1:oncall
```

> **Gotcha:** With the default `missing`, an error-count alarm on a quiet function sits in `INSUFFICIENT_DATA`, because no errors means no data points. For sparse error metrics, treat missing data as `notBreaching`.

Dashboards and service defaults

Service	Logs	Key metrics
Lambda	`/aws/lambda/<fn>`	`Errors`, `Throttles`, `Duration`
ECS	`awslogs` driver group	`CPUUtilization`, `MemoryUtilization`
ECS + Container Insights	Performance log events	Task counts, storage, network

Lambda writes a `REPORT` line per invocation with duration and memory used, which is why the Insights example above works with no code. ECS service metrics carry `ClusterName` and `ServiceName` dimensions; Container Insights adds per-service task counts and more. A dashboard is a JSON document of widgets — keep it in IaC next to the alarms.

```
aws cloudwatch get-dashboard \
  --dashboard-name api \
  --query DashboardBody --output text \
  > dashboard.json
```

Cost traps

Trap	Fix
High-cardinality dimensions	IDs in log fields, never in dimensions
Verbose ingestion	Log levels; Infrequent Access class
Infinite retention	Retention on every group
Wide Insights scans	Short ranges, `limit any`, fewer groups

Log ingestion is billed per GB as it arrives, so debug logging left on in production is paid for on every line, before anyone reads it.

`limit any N` stops scanning once it has N matches, which helps when you need one example, not a sorted list.

```
fields @message
| filter @message like /Timeout/
| limit any 1
```

> **Tip:** For distributed tracing, use X-Ray or OpenTelemetry rather than reconstructing request paths from logs — see the OpenTelemetry sheet.

Further reading

- CloudWatch Logs User Guide
- Logs Insights query syntax
- Embedded metric format specification
- Amazon CloudWatch pricing