Monitoring Guide

1. Understanding Anomaly Scores

The anomaly score (0–10) represents how many standard deviations a value is from the learned baseline mean.

- **Score 1–2**: Minor fluctuation. Usually noise.
- **Score 3–4**: Moderate deviation. Worth investigating if persistent.
- **Score 5–7**: Significant anomaly. Likely a real change in behavior.
- **Score 8–10**: Extreme outlier. Immediate attention recommended.

2. Choosing the Right Threshold

The threshold controls sensitivity. Lower = more alerts, higher = fewer but more significant.

- **1.0–2.0**: Testing only. Extremely noisy. Catches normal variance.
- **3.0** (default): Catches top 0.3% outliers. Good starting point for production.
- **4.0–5.0**: Reduced noise. Recommended for naturally variable metrics (Duration, CPU).
- **6.0+**: Only extreme spikes/drops. Use for metrics where small changes are expected.
- **-1**: Z-score alerting disabled. Only "Alert if below" / "Alert if above" static thresholds will fire.

3. Direction Setting

Controls which direction of deviation triggers alerts.

- **Both**: Alert on spikes AND drops. Use for metrics where either direction is concerning.
- **High**: Only alert when value is ABOVE mean. Best for: Duration, Errors, CPU, Latency, Throttles.
- **Low**: Only alert when value is BELOW mean. Best for: FreeStorage, SuccessRate, Throughput, HealthyHosts.

4. Example Settings by Metric Type

Metric	Threshold	Direction	Notes
Lambda/Duration	4.0	High	Cold starts cause natural spikes; use higher threshold
Lambda/Errors	3.0	High	Any error spike is significant
Lambda/Throttles	3.0	High	Indicates capacity issues
Lambda/ConcurrentExec	4.0	High	Spikes indicate load; drops are normal
ES/CPUUtilization	5.0	High	CPU varies naturally; only care about sustained highs

ES/IndexingRate	4.0	Low	Drop means ingestion pipeline may be broken
SQS/MessageAge	3.0	High	Growing age means processing is falling behind
SQS/MessagesVisible	4.0	High	Queue buildup indicates downstream issues
Firehose/DataFreshness	3.0	High	Delivery lag indicates pipeline problem

5. Wildcard Dimensions (Regex)

Use regex in dimension values to monitor multiple resources with one subscription.

- `{"FunctionName": ".*"}` – Monitor ALL Lambda functions independently
- `{"FunctionName": "appa-.*"}` – Only functions starting with "appa-"
- `{"FunctionName": "(AppA|AppB)"}` – Specific functions by name
- `{"DomainName": "logs-.*", "NodeId": ".*"}` – Fixed domain, wildcard each node

Each unique dimension combination gets its own baseline model. A spike in one function does not affect another's scoring.

Tip: Use the ✨ sparkle button to wildcard all dimensions at once, then manually fix the ones you want exact.

6. Reducing False Positives

- **Raise threshold** to 4.0–5.0 for naturally variable metrics
- **Set direction** appropriately (high-only for errors/duration, low-only for availability)
- **Wait for baseline** – the first 6–24 hours will be noisy as the model learns
- **Narrow dimensions** – monitor specific resources instead of account-wide aggregates
- **Use maintenance windows** – suppress alerts during known deploy/maintenance periods
- **Disable noisy subscriptions** temporarily while tuning (double-click in list to toggle)
- **Consider Alert if …** when there is a hard breach (e.g. free disk below)

7. Baseline Learning Period

The detector needs a minimum number of datapoints before scoring begins.

- Default: 5% of 288-point window = ~15 datapoints
- At 5-minute collection intervals: ~75 minutes before first alert
- The baseline window holds 288 points (24 hours at 5-min intervals)
- Older data rolls off automatically – the model adapts to new normals over time

Note: If a metric's behavior genuinely changes (e.g., you scaled up), the baseline will adapt

within 24 hours and stop alerting on the new normal.

8. Trend Indicators

The subscription list shows trend arrows based on anomaly frequency:

- ▲ **Worsening**: More anomalies this week than last week
- ▼ **Improving**: Fewer anomalies this week than last week
- ▶ **Stable**: Same frequency
- ★ **New**: Anomalies detected but no prior week data for comparison

A "worsening" trend over multiple weeks may indicate a systemic issue that needs investigation beyond individual alert triage.

9. Maintenance Windows

Create maintenance windows to suppress alerts during planned events:

- **Scheduled**: Recurring (e.g., every Tuesday 06:00–06:30 UTC for weekly deploys)
- **Ad-hoc**: One-time, starts immediately for a set duration

Select which subscriptions are affected. Use "All" for full-system maintenance.

When a window is active, anomalies are still detected and recorded but alerts are suppressed. An SNS notification is sent when a window starts/ends.

10. Account ID Field

Controls which AWS accounts are monitored for this subscription:

- **Blank/local only**: Only metrics from the local account (default)
- *** (all)**: Metrics from all accounts (local + cross-account)
- **Specific IDs**: Select one or more accounts from the dropdown

Cross-account monitoring requires the CrossAccountIds stack parameter to be configured and remote accounts to have Firehose delivery set up.

11. Common Questions

Q: Why did I get an alert for a normal value?

A: The baseline is still learning. Wait 6–24 hours, or raise the threshold. If the value is consistently "normal" at that level, the baseline will adapt and stop alerting.

Q: Why are dimensions showing "none"?

A: The subscription has empty dimensions `{}` meaning it monitors the account-wide aggregate. Consider adding specific dimensions or using `{"FunctionName": ".*"}` for per-resource monitoring.

Q: Can I monitor a metric that doesn't exist yet?

A: Yes. The subscription will wait until data arrives. Once the Metric Stream delivers datapoints for that namespace/metric, collection begins automatically.

Q: How do I stop getting flooded with emails?

A: The detector sends at most one alert per subscription per 5-minute cycle. To reduce further: raise thresholds, set correct direction, or disable noisy subscriptions.

Q: What happens when I change a subscription's settings?

A: Changes take effect on the next detection cycle (within 5 minutes). The baseline model is preserved – it doesn't reset unless you delete and recreate the subscription.

Q: What does the sparkle button do?

A: It replaces all dimension values with `.*` (wildcard), meaning "monitor each unique resource independently." You can then manually set specific values for dimensions you want fixed.

Q: How does AI Monitor learn? Does it understand daily/weekly/seasonal patterns?

A: AI Monitor uses a sliding window of the last 288 datapoints (approximately 24 hours at 5-minute intervals). It learns what "normal" looks like based on recent history:

- **Day-to-day:** The 24-hour window naturally captures a full daily cycle. If your metric is higher during business hours and lower overnight, the baseline's mean and standard deviation reflect that combined pattern. However, it does not distinguish "2pm normal" from "3am normal" – it uses one aggregate baseline.
- **Week-to-week:** The window is 24 hours, not 7 days. It does NOT learn that "Mondays are busier than Sundays." If Monday traffic spikes compared to the Sunday-dominated baseline, it may trigger. However, by Tuesday the baseline has adapted to the new weekday level.
- **Season-to-season:** Not currently supported, let us know if interested. Gradual long-term growth (e.g., traffic doubling over 3 months) is handled because the sliding window continuously adapts – old values drop off and the mean shifts. But sudden seasonal events (e.g., Black Friday) will trigger anomalies unless you set up a maintenance window.

How adaptation works: As new datapoints arrive, the oldest ones are dropped from the window. This means:

- A genuine step-change (e.g., you deployed a new version that's 2x faster) will cause alerts for ~24 hours, then the baseline adapts and alerts stop.
- A spike that returns to normal will be scored high, but won't pollute the baseline significantly (one point among 288).
- Gradual drift is invisible to the detector – the window moves with it.

Limitations and future improvements:

- No time-of-day awareness (planned: separate weekday/weekend baselines)
- No week-over-week comparison for scoring (planned: multi-scale windowing)
- No seasonality decomposition (e.g., holiday patterns)

Workaround for weekly patterns: If you know Monday mornings always spike, create a scheduled maintenance window for that period. Or raise the threshold to accommodate the variance.