Introduction

As part of strengthening our monitoring and observability, I conducted a gap analysis between the metrics we currently monitor in CloudWatch and the recommendations provided by AWS. The goal was not just to tick boxes but to focus on **metrics that give us actionable insights** — the kind that tell us *what to do next* if a threshold is crossed.

This document captures where we are today, what AWS suggests, which metrics actually add value, and which ones we can safely deprioritize.

Current Monitoring

Our dashboards already track some of the basics — CPU utilization, freeable memory, disk usage, network throughput, and database connections. These give us a general health view, but as we all know, **not every metric is equally useful**. For example, we've been watching *Freeable Memory*, but in reality this metric doesn't drive any concrete action. AWS confirms this in their guidance.

Where AWS Recommendations Add Value

AWS highlights three metrics as the most critical for actionable database performance monitoring:

1. **CPU Utilization** – This is already in place. Sustained high CPU tells us when to scale up or rebalance queries across replicas. One improvement we can make is to break this down by replica instead of just looking at the cluster average.
2. **BufferCacheHitRatio** – This one we are currently missing. A consistent value below 99% indicates that our instance doesn't have enough memory to cache the working set. In practice, this means the database is forced to go to disk too often, which degrades performance. Adding this metric will help us right-size memory for the workload.
3. **MainRequestQueuePendingRequests** – Also missing today. Anytime this is greater than zero, it means queries are piling up because there aren't enough worker threads. This is a very clear sign that we either need to scale up or optimize query concurrency. It's a metric that speaks directly to user experience, so it should be added.

Other Metrics from AWS

In addition to the “big three,” AWS suggests:

- **OpenCypherBoltOpenConnections / GremlinWebSocketOpenConnections** – These help us detect client connection leaks. If we are using Bolt or Gremlin, this would be valuable. If not, we can skip it.
- **GremlinClientErrorsPerSec / NumOpenCypherClientErrorsPerSec / TotalClientErrorsPerSec** – These give us visibility into client-side errors. The key here is to baseline what “normal” looks like, then set alerts for any significant deviation.
- **Event Subscriptions** – AWS strongly recommends enabling this for control plane notifications (backups, failovers, maintenance events). This isn’t in place yet, and it’s a simple win.

Metrics We Should Deprioritize

- **Freeable Memory** – While we’ve been tracking this, it doesn’t tell us much. A drop doesn’t necessarily mean a problem; caching and workload patterns can explain fluctuations. We’ll keep it for visibility but remove it from alerting.
- **Basic Network Traffic (In/Out)** – We already have more precise metrics through connection counts and query performance. Alerts on raw network throughput don’t add real value.
- **Unstructured Error Counts** – Logging error volume without categorization creates noise. We’ll rely instead on the structured client error metrics mentioned above.

Additional Metrics We Should Add Beyond AWS Guidance

Through experience, I believe there are a few more metrics that would bring real operational value:

- **Query Latency (P50/P95/P99)** – This gives us a user-focused view of how queries are performing. Spikes in P95/P99 latency usually indicate emerging performance bottlenecks.
- **Replica Lag** – Critical for ensuring read replicas are fresh. Any consistent lag undermines data consistency guarantees.
- **Failover Events** – We should explicitly track role changes to make sure applications are adapting as expected.
- **Backup Success Rate** – For compliance, we need visibility into whether automated backups and snapshots succeed.
- **IAM Login Failures / Unauthorized API Calls** – From a security angle, this ensures we have early warning of brute force attempts or misconfigured apps.

Recommendations

- **Add** BufferCacheHitRatio, MainRequestQueuePendingRequests, Client Error Metrics, and (if applicable) Connection Leak Metrics.
- **Enable** Event Subscriptions for Neptune control plane.
- **Deprioritize** Freeable Memory alerts and raw network traffic metrics.
- **Enhance** dashboards with latency, replica lag, failover, backup, and security-related metrics.
- **Document** thresholds and corrective actions in Confluence so alerts tie directly into runbooks.

Conclusion

Not all metrics are created equal. By focusing on those that are **actionable**, we keep dashboards clean and meaningful. Adding the missing AWS-recommended metrics, enhancing with latency and reliability indicators, and pruning non-actionable ones will put us in a much better place to detect issues early, scale appropriately, and stay compliant.