Observability Maturity Model for AWS – From Reactive to Autonomous

Indika Wimalasuriya

Cloud Native 2024

Agenda

- Importance of Observability in Cloud-Native
- Need for Observability Maturity Model
- Pillars of AWS Observability
- 4 Stages of AWS Observability Maturity Model
- Implementation Guidelines
- Progress Measurement with Business Outcomes
- Best Practices and Pitfalls
- Future of Cloud-Native Observability

Why Observability Matters for Cloud-Native

Distributed System Complexity	Dynamic and Elastic Nature	Container Orchestration
Continuous Deployment and Integration	Service Mesh and API gateways	Increased Complexity of Cloud Services
Scalability Challenges	Quick Detection and Resolution of Issues	End to End Visibility

Why You Need Observability Maturity Model for Cloud-Native?

Pillars Shaping AWS Observability

Logs - Records of events and activities in your systems and applications. Useful for troubleshooting issues and auditing.

Metrics - Quantitative data about performance and behavior over time. Help track trends and identify anomalies.

Tracing - Follows a request as it flows through distributed systems. Used to analyze bottlenecks and errors.

Alarms - Automated notifications when certain thresholds are breached. Help quickly identify and respond to issues.

Dashboards - Visual representations of metrics, logs and other data. Provide at-a-glance views of system health.

Canaries - Automated tests that run synthetic transactions to monitor availability and performance

Real User Monitoring - Captures performance from an end user perspective. Surfaces issues that affect users.

Infrastructure Monitoring - Monitors the health and utilization of underlying resources like servers, databases, etc.

Network Monitoring - Observes network connectivity and traffic to detect problems and optimize performance.

Security Monitoring - Detection of security threats, anomalies and unauthorized activities.

Cost Optimization - Tracking usage and spending to optimize costs.

The 4 Stages of the AWS Observability Maturity Model

Reactive level

- Logs used for incident troubleshooting.
- Minimal proactive monitoring.
- Limited metrics collection.

Proactive Level

- Logs monitored with alerts.
- Basic metrics collected and monitored.
- Threshold-based monitoring implemented.
- Initial steps towards predictive insights.

Predictive Level

- Advanced trend prediction analytics.
- Distributed tracing for optimization.
- Canaries for predictive insights.
- Automated anomaly detection.
- Real-time user monitoring.
- Proactive issue prevention.

Autonomous Level

- Automated analysis, correlation, anomaly detection.
- Self-adaptive canaries, infrastructure.
- Proactive scaling, resource management.
- Real-time user monitoring, automated remediation.
- Predictive analytics for optimization.
- Continuous improvement, self-optimization.
- Integration with CI/CD pipelines.
- AI-driven decision-making.

AWS Pillars from Reactive to Autonomous

Pillars of AWS Observability	Reactive	Proactive	Predictive	Autonomous
Logs	• Logs used for troubleshooting after incidents	• Monitoring logs with alerts for abnormal patterns	• Advanced analysis for trend prediction	• Automated analysis, correlation, anomaly detection
Metrics	• Basic collection, not actively monitored	• Monitoring metrics with predefined thresholds	• Advanced analytics for anomaly detection	• Automated scaling, anomaly detection based on ML
Tracing	• Tracing not implemented	• Basic tracing for critical services	• Distributed tracing for performance optimization	• Automated tracing, root cause analysis
Canaries	• Canaries not utilized	• Basic canaries for critical services	• Advanced canaries for predictive insights	• Self-adaptive canaries, automatic scaling
Real User Monitoring (RUM)	• RUM data not collected	• Basic RUM data collection for user experience	• Advanced analytics for predicting user behavior	• Automated optimization based on RUM and analytics

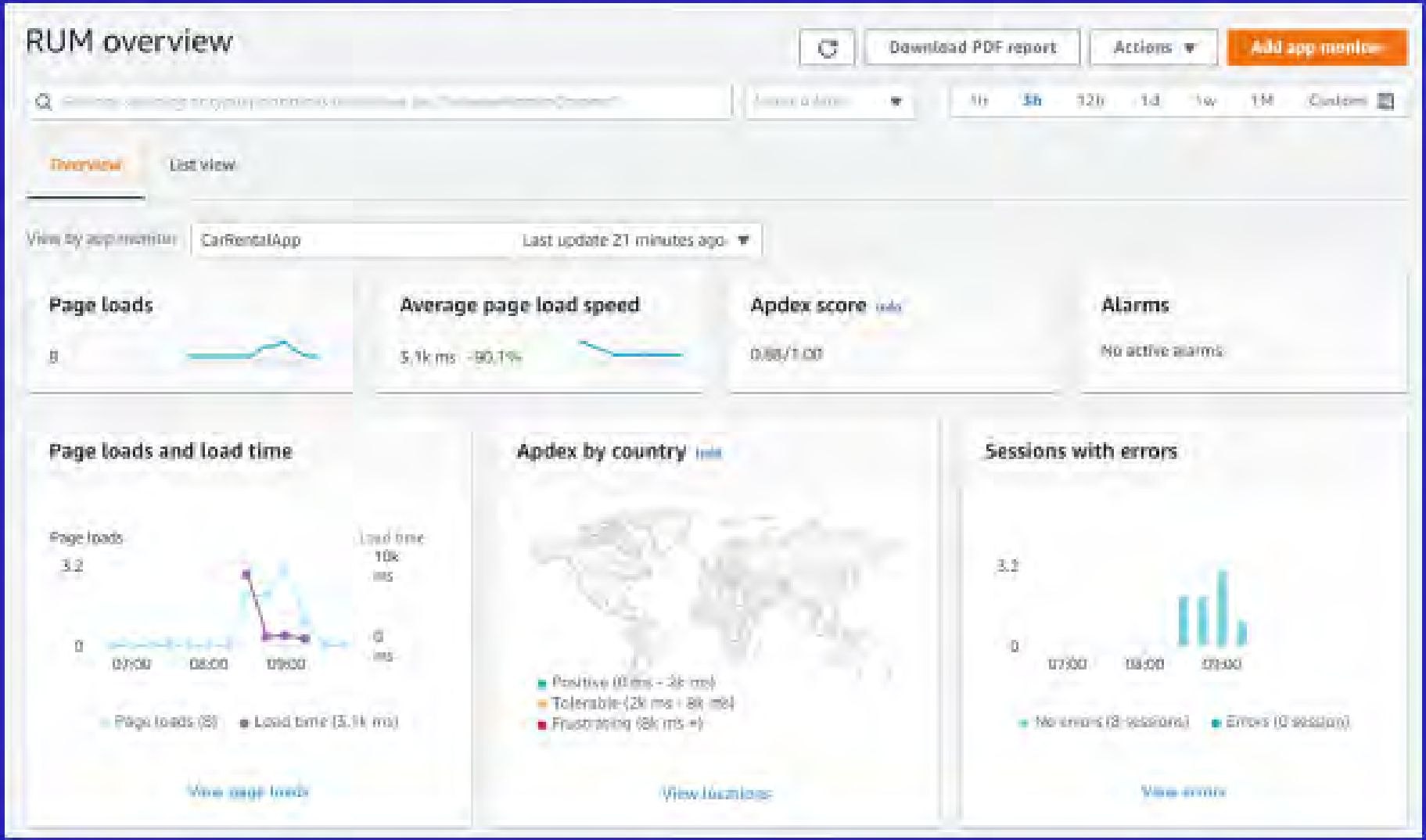
AWS Pillars from Reactive to Autonomous (Cont.)

Pillars of AWS Observability	Reactive	Proactive	Predictive	Autonomous
Infrastructure Monitoring	• Basic metrics collected, not actively monitored	• Automated monitoring with alerts for deviations	• Predictive maintenance and capacity planning	• Self-healing infrastructure, automated scaling
Network Monitoring	• Network monitoring tools not implemented	• Basic network monitoring for outages/performance	• Advanced analytics for security threats	• Self-adaptive network monitoring, dynamic config
Security Monitoring	• Security monitoring not implemented	• Basic monitoring tools for known threats	• Real-time threat detection, automated response	• Autonomous security monitoring with AI
Cost Optimization	• Cost optimization not considered	• Basic strategies based on manual analysis	• Advanced optimization using automation/predictive	• Fully automated cost optimization, CI/CD integrated

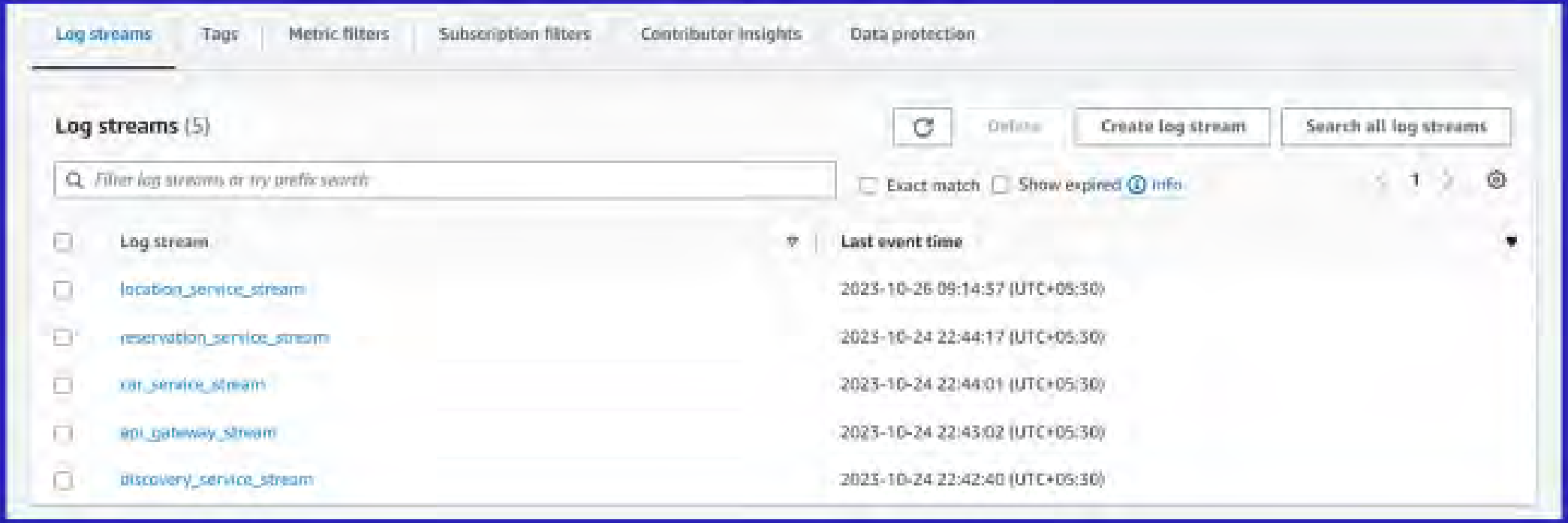
Implementation Guidelines

- Real User Monitoring (RUM)
- Application Performance Monitoring (APM)
- Distributed Tracing
- Logs & Events
- Metrics & SLOs
- Infrastructure Monitoring

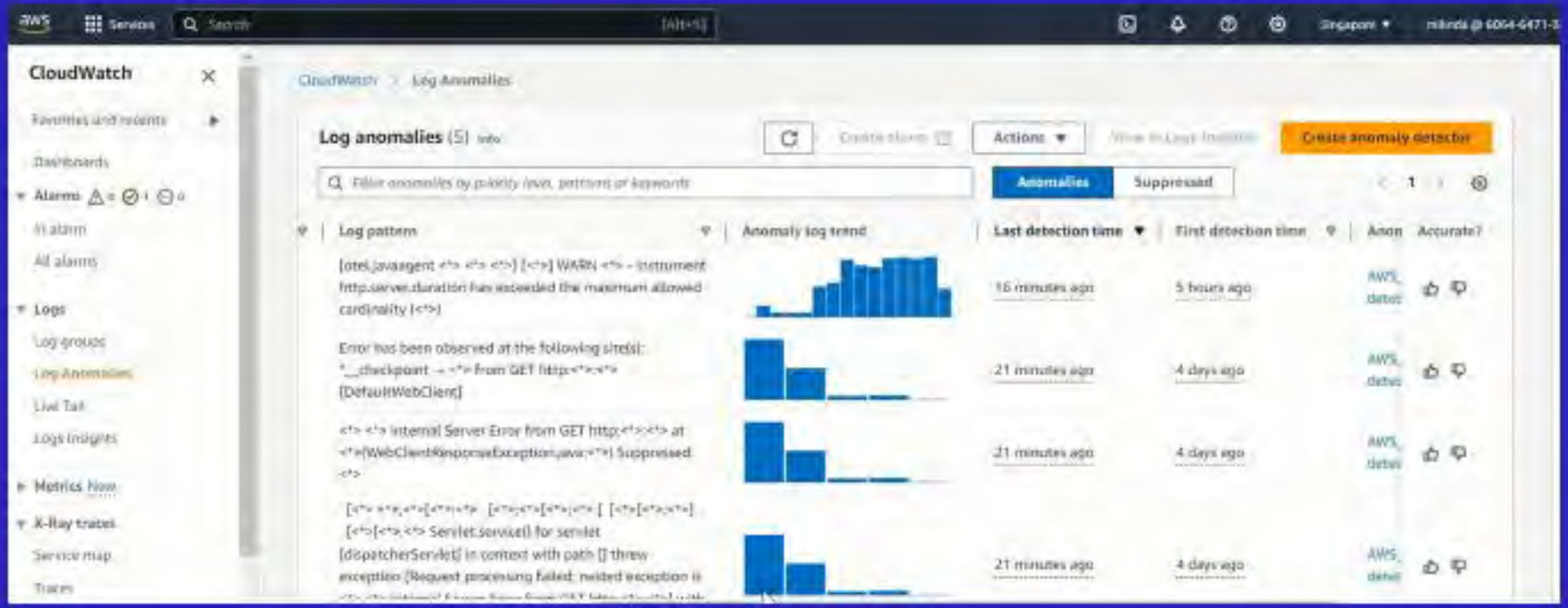
Enable Real User Monitoring (RUM)



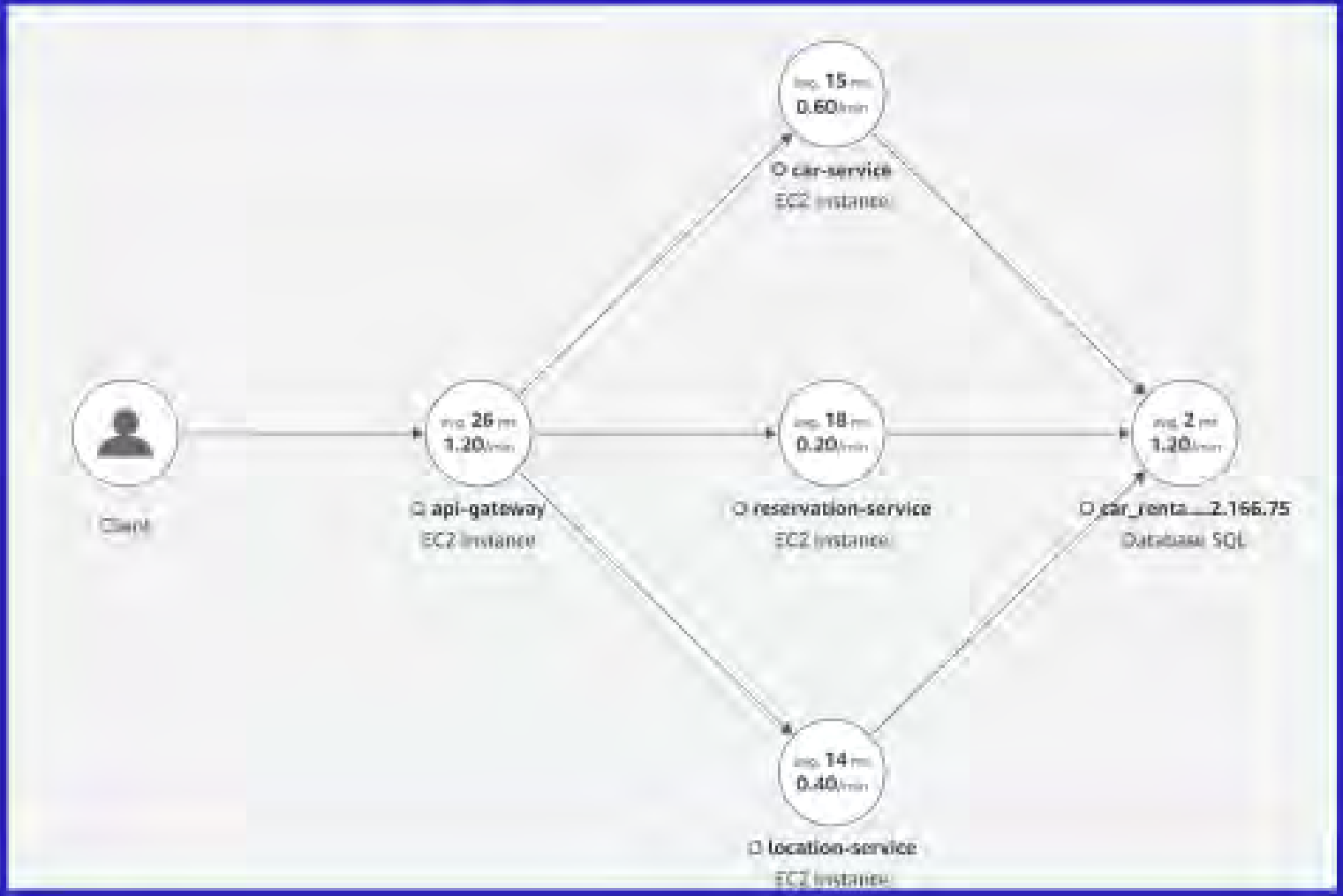
Enable Log Monitoring



Enable Log Anomaly Detection

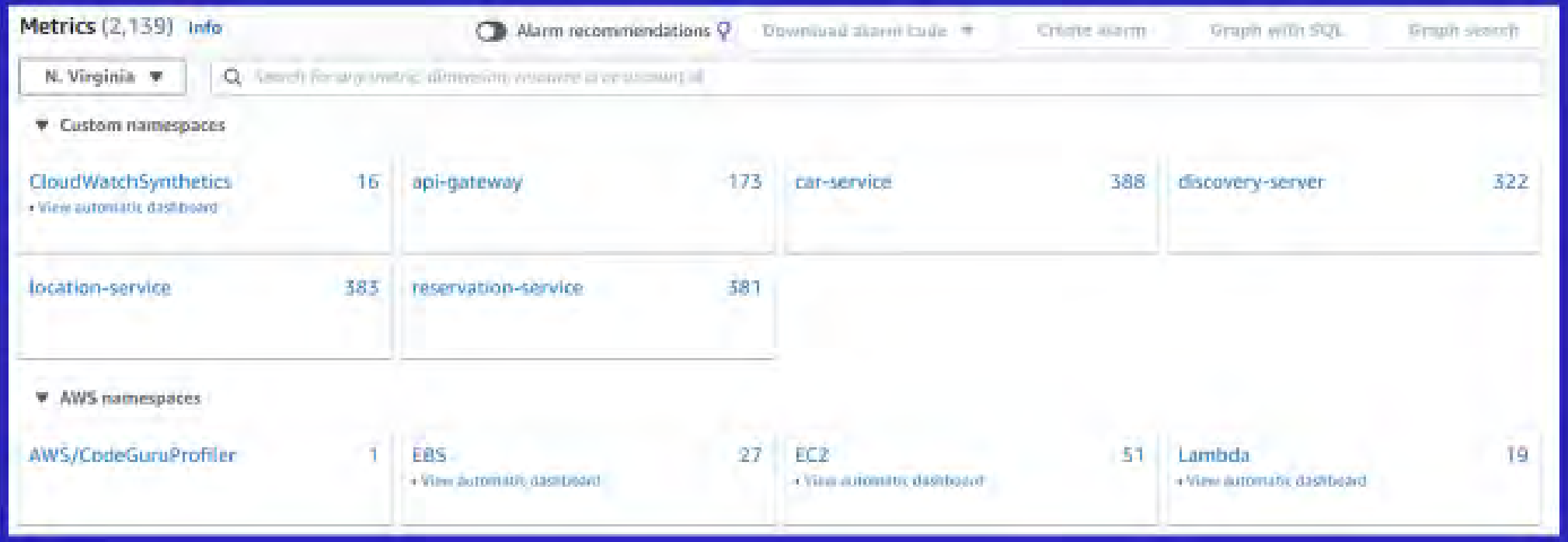


Enable Tracers with OTEL

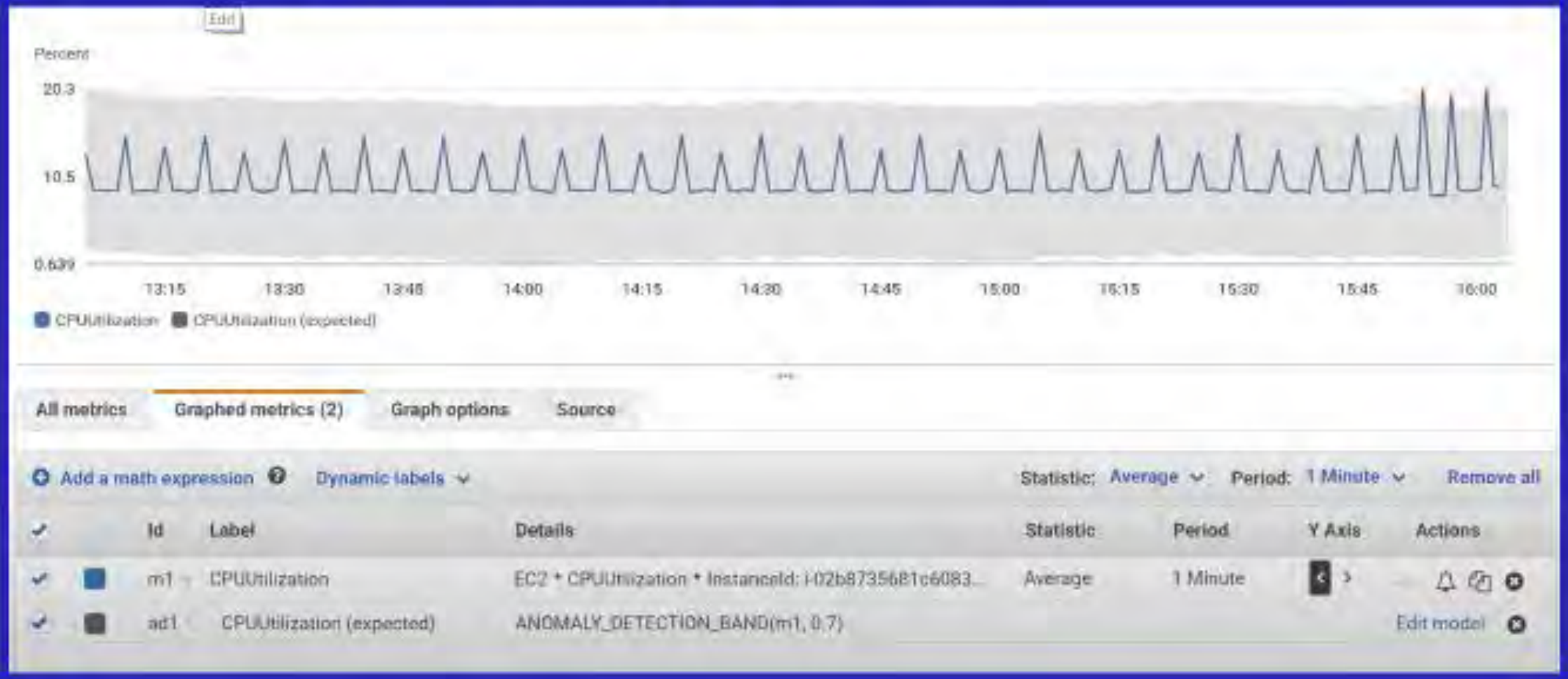


Enable Tracers with OTEL

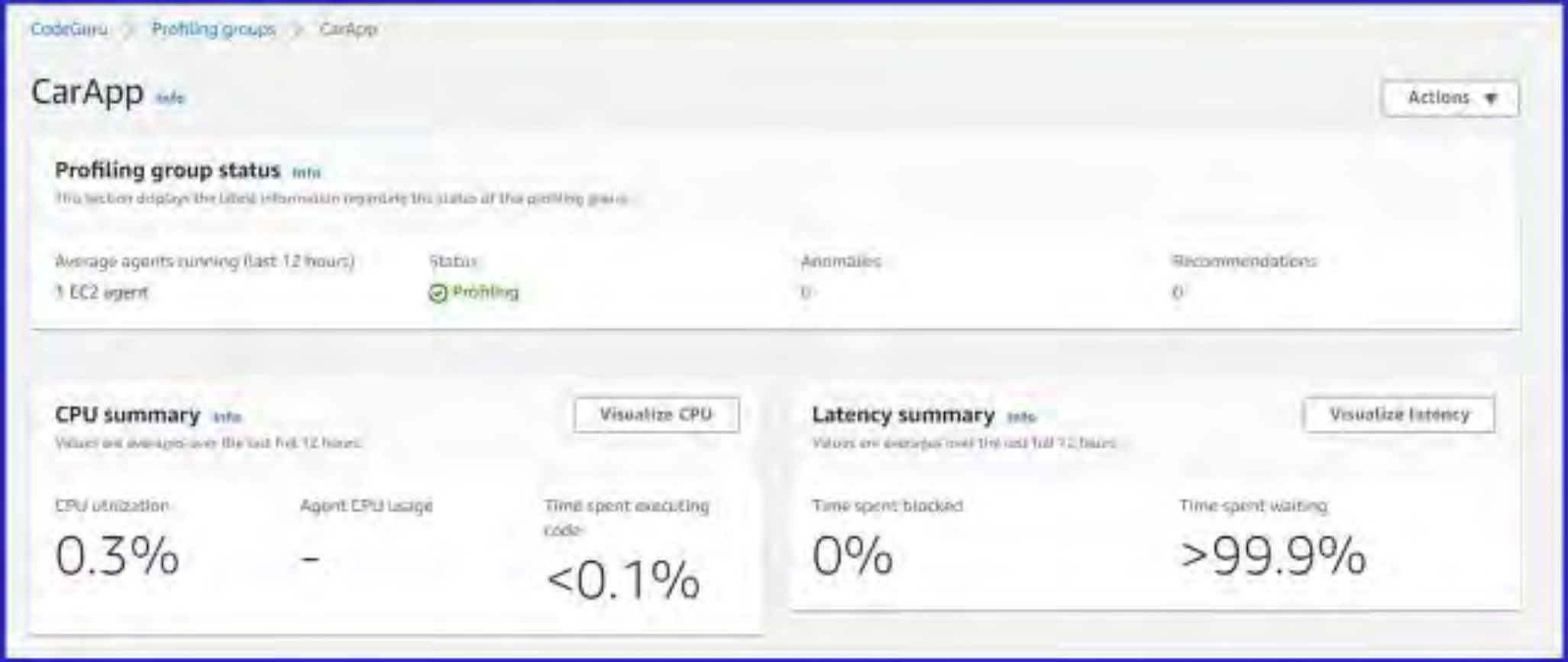
Enable Metrics



Enable Metric Anomaly Detection



Enable CodeGuru for Profiling



Enable DevOps Guru

Amazon DevOps Guru > Analyzed resources

Analyzed resources for Account 606464713227

Edit analyzed resources

Refreshed on October 26, 2023 03:38 UTC

Applications analyzed Info

Number of applications
2

Coverage selection type
All account resources

Summary

Service types analyzed
3

Dimensions analyzed
3

Analyzed resources (3)

Download CSV resource list

DevOps Guru charges based on the number of AWS resource types analyzed, such as per SQS queue or per DynamoDB table. Tags can be used to select specific AWS resources analyzed by DevOps Guru.

Search by resource, stack name, service name

1

Resource name	Service name	Stack name
cwsyn-runbasictest-a1555c8e-75a4-4...	Lambda	-
database-1	RDS	-
Default_CloudWatch_Alarms_Topic	SNS	-

Measure Progress with Business Outcomes

- Define clear goals (e.g., reducing downtime, enhancing satisfaction).
- Track observability's impact over time.
- Focus on metrics like cost reduction, faster issue resolution.
- Set improvement targets for each maturity stage.
- Use data to quantify customer experience benefits.
- Relate maturity stages to enhanced customer service.
- Showcase how maturity accelerates innovation.
- Align observability with strategic customer-focused objectives.
- Secure executive buy-in by highlighting customer-centric results.

Best Practices

- Use CloudWatch for metrics, logs, and alarms. CloudWatch provides a centralized place for monitoring across AWS services.
- Enable enhanced monitoring for EC2, ELB, RDS, etc. The detailed metrics can help troubleshoot issues.
- Use X-Ray for distributed tracing. X-Ray helps trace requests across services and identify latency issues.
- Send application logs to CloudWatch Logs or third-party tools. Centralized logging is critical for debugging errors.
- Set up dashboards in CloudWatch. Pre-built and custom dashboards provide visibility into key metrics.

Pitfalls to Avoid

- Not retaining logs and metrics for an adequate period, losing historical visibility.
- Just monitoring at a high-level system view, lacking more granular insights.
- Failing to set up alarms and notifications, missing early awareness of problems.
- Not monitoring all critical services and resources, leading to blindspots.
- Having observability data siloed across tools, lacking centralized visibility.

Where Cloud Native Observability Is Heading?

Immediate Future:

- Increased adoption of OpenTelemetry.
- More observability tools supporting serverless and containers.
- Tighter integration between monitoring, logging, and tracing.

Mid-Term Outlook:

- Automated root cause analysis and alert correlation.
- Rise of AIOps using ML for telemetry analysis.
- Native observability features in cloud platforms.
- Shift to logs-based monitoring in cloud-native apps.

Long-Term Vision:

- Observability embedded into dev workflows and GitOps.
- Adoption of adaptive, real-time alerting.
- Observability data driving automated remediation.
- Rise of predictive monitoring powered by ML.
- Integration of observability across organizational silos.

Will Amazon Web Services break into the Leader level in Gartner's Observability Magic Quadrant?

Thank you!