Day 17: Setting Alarms, Dashboards, and Metrics

AWS CloudWatch - Monitoring Advanced

Learning Objectives

- Create and manage CloudWatch Alarms
- Build custom Dashboards
- Publish and use custom CloudWatch Metrics
- Trigger actions based on metric thresholds

Why Use Alarms, Dashboards, and Metrics?

- Alarms: Auto-heal or alert when thresholds are breached
- Dashboards: Visualize performance and trends
- Custom Metrics: Application-specific observability

Security and IAM Notes

To create alarms, push metrics, and create dashboards, users need permissions like:

cloudwatch:PutMetricAlarm
cloudwatch:GetMetricData
cloudwatch:PutDashboard
sns:Publish (for notifications)

Alarm States and Types

- States: OK, ALARM, INSUFFICIENT_DATA
- Static Alarms: Based on fixed thresholds
- Anomaly Detection: Detect deviation from patterns

Lab 1: Create a Static Alarm

- Go to CloudWatch > Alarms > Create
- Select EC2 > CPUUtilization > Threshold: >70%
- Add SNS topic for alerting
- Test using stress tool on EC2

Lab 2: Create Anomaly Detection Alarm

- Select EC2 metric (e.g., NetworkOut)
- Enable anomaly detection
- Configure actions for ALARM/OK
- Simulate traffic to test

CloudWatch Dashboards

- Create from CloudWatch > Dashboards
- Add widgets: graphs, gauges, numbers
- Use for EC2 CPU, Network metrics
- Optional: Add text descriptions

Lab 3: Dashboard Creation

- Create dashboard: MyEC2Dashboard
- Add: CPUUtilization, NetworkIn/Out graphs
- Add text box for notes
- Observe live data

Custom Metrics

- Use CLI to push data: put-metric-data
- Namespace: Custom/MyApp
- View in CloudWatch > Metrics > Custom
- Create alarms on these metrics

Lab 4: Push Custom Metric

- Use CLI to push value to CloudWatch
- Command: aws cloudwatch put-metric-data
- Create alarm on custom metric
- Verify appearance in Metrics dashboard

Commands Summary

- Create alarm (CLI): aws cloudwatch put-metric-alarm ...
- Push custom metric: aws cloudwatch put-metric-data ...
- Stress test: stress --cpu 1 --timeout 300
- Navigate: CloudWatch > Metrics / Dashboards

Best Practices

- Use SNS topics for scalable notifications
- Apply anomaly detection to dynamic metrics
- Keep dashboards focused and readable
- Use custom metrics for app-level observability

Summary & What's Next

- Alarms enable proactive monitoring
- Dashboards give real-time visibility
- Custom metrics track app-specific behavior