SRE Dashboards with AWS EC2, CloudWatch Metrics & Logs

5-Hour Hands-On Workshop with Log Analysis

Workshop Overview

Duration: 5 hours
Level: Intermediate
Prerequisites: AWS account with EC2, CloudWatch, and CloudWatch Logs access

Learning Objectives:

- Configure CloudWatch monitoring for EC2 instances
- Set up CloudWatch Logs for application logging
- Create SRE dashboards with metrics and log insights
- Write CloudWatch Logs Insights queries
- Set up alarms and notifications
- Implement SRE metrics (SLIs, SLOs)

Part 1: SRE Concepts Overview

Key Metrics to Monitor:

- **Latency:** Response time of requests
- **Traffic:** Number of requests per second
- **Errors:** Failed request rate
- **Saturation:** CPU, Memory, Disk usage

SRE Targets:

- **SLI:** Availability percentage (e.g., 99.9%)
- **SLO:** Target uptime (e.g., 99.9% = 43 minutes downtime/month)
- **Error Budget:** Remaining allowable downtime

Part 2: Create IAM Role for CloudWatch

⚠️ CRITICAL: Do this BEFORE launching EC2 instances!

Step 1: Navigate to IAM Console

1. Open AWS Console
2. Search for "IAM" in top search bar
3. Click IAM service

Step 2: Create IAM Role

1. Click **Roles** in left navigation menu
2. Click **Create role** button
3. Select trusted entity type: **AWS service**
4. Use case: Select **EC2**
5. Click **Next**

Step 3: Attach Policies

1. In the search box, type: `CloudWatchAgentServerPolicy`
2. Check the box next to **CloudWatchAgentServerPolicy**
3. Search for: `CloudWatchLogsFullAccess`
4. Check the box next to **CloudWatchLogsFullAccess**
5. (Optional) Also add: `AmazonSSMManagedInstanceCore` for better management
6. Click **Next**

Step 4: Name and Create Role

1. Role name: `SRE-Workshop-CloudWatch-Logs-Role`
2. Description: `Allows EC2 instances to send metrics and logs to CloudWatch`
3. Click **Create role**

✅ **You now have an IAM role ready for metrics and logs!**

Part 3: Launch EC2 Instances

Step 1: Navigate to EC2 Console

1. Open AWS Console
2. Search for "EC2" in top search bar
3. Click EC2 service

Step 2: Launch Instance

1. Click **Launch Instance** button (orange)

2. Name: `SRE-Workshop-Web-01`
3. Select **Amazon Linux 2023 AMI**
4. Instance type: `t3.micro`
5. Key pair: Select existing or create new
6. Network settings: Allow SSH (port 22) and HTTP (port 80)
7. ⚠️ **IMPORTANT:** Expand **Advanced details** section
8. **IAM instance profile:** Select `SRE-Workshop-CloudWatch-Logs-Role`
9. Click **Launch Instance**
10. Repeat for second instance: `SRE-Workshop-Web-02`

✅ **Both instances now have permission to send metrics and logs to CloudWatch!**

Step 3: Add Tags

1. Select instance
2. Click **Tags** tab
3. Click **Manage tags**
4. Add tags:
    - Key: `Environment`, Value: `Workshop`
    - Key: `Application`, Value: `WebServer`
    - Key: `Team`, Value: `SRE`
5. Click **Save**

Step 4: Connect to Instance

1. Select instance
2. Click **Connect** button
3. Choose **EC2 Instance Connect** tab
4. Click **Connect** button
5. Browser terminal opens

Step 5: Install CloudWatch Agent

Execute these commands in terminal:

```bash
sudo dnf update -y
sudo dnf install amazon-cloudwatch-agent -y
```

Part 4: Create CloudWatch Agent Configuration with Logs

Step 1: Create Configuration File

```bash
sudo nano /opt/aws/amazon-cloudwatch-agent/etc/config.json
```

Step 2: Paste Configuration with Logs

Copy and paste this configuration (includes metrics AND logs):

```json
```

```json
{
 "agent": {
  "metrics_collection_interval": 60,
  "run_as_user": "root"
 },
 "logs": {
  "logs_collected": {
   "files": {
    "collect_list": [
     {
      "file_path": "/var/log/nginx/access.log",
      "log_group_name": "/aws/ec2/sre-workshop/nginx/access",
      "log_stream_name": "{instance_id}",
      "retention_in_days": 7,
      "timezone": "UTC"
     },
     {
      "file_path": "/var/log/nginx/error.log",
      "log_group_name": "/aws/ec2/sre-workshop/nginx/error",
      "log_stream_name": "{instance_id}",
      "retention_in_days": 7,
      "timezone": "UTC"
     },
     {
      "file_path": "/var/log/messages",
      "log_group_name": "/aws/ec2/sre-workshop/system",
      "log_stream_name": "{instance_id}",
      "retention_in_days": 7,
      "timezone": "UTC"
     }
    ]
   }
  }
 },
 "metrics": {
  "namespace": "SREWorkshop",
  "metrics_collected": {
   "cpu": {
    "measurement": [
     "cpu_usage_idle",
     "cpu_usage_iowait"
    ],
    "totalcpu": false
   },
   "disk": {
    "measurement": [
          "used_percent"
        ],
        "resources": [
          "*"
        ]
      },
      "mem": {
        "measurement": [
          "mem_used_percent"
        ]
      }
    }
  }
}
```

Press `Ctrl+X`, then `Y`, then `Enter` to save

Step 3: Start CloudWatch Agent

```bash
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -m ec2 -s -c file:/opt/aws/amazon-
```

Expected output: `Configuration validation succeeded`

Step 4: Verify Agent Status

```bash
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -m ec2 -a status -c default
```

Expected output:

```json
{
  "status": "running",
  "starttime": "2025-10-05T11:34:53+00:00",
  "configstatus": "configured",
  "version": "1.300057.2"
}
```

Also verify with systemd:

```bash
```

```
systemctl status amazon-cloudwatch-agent
```

Expected output: Should show `active (running)` in green

Step 5: Wait for Metrics and Logs to Appear

⏱️ **IMPORTANT:** Wait 3-5 minutes for metrics and logs to appear in CloudWatch!

- The agent collects metrics every 60 seconds
- Logs are sent in near real-time
- Log groups will be created automatically

Part 5: Install Sample Application

Step 1: Install Web Server

```bash
sudo dnf install nginx -y
sudo systemctl start nginx
sudo systemctl enable nginx
```

Step 2: Verify Installation

```bash
curl localhost
```

Step 3: Generate Some Traffic

```bash
for i in {1..100}; do curl localhost; done
```

Step 4: Check Nginx Status

```bash
sudo systemctl status nginx
```

Step 5: View Nginx Logs

```bash
```

```
sudo tail -f /var/log/nginx/access.log
```

Press `Ctrl+C` to stop viewing logs

Part 6: Verify Log Groups Created

Step 1: Navigate to CloudWatch Logs

1. Open AWS Console (new browser tab)
2. Search for "CloudWatch" in top search bar
3. Click CloudWatch service
4. Click **Logs** → **Log groups** in left navigation menu

Step 2: Verify Log Groups

You should see three log groups created:

- `/aws/ec2/sre-workshop/nginx/access`
- `/aws/ec2/sre-workshop/nginx/error`
- `/aws/ec2/sre-workshop/system`

Step 3: View Log Streams

1. Click on `/aws/ec2/sre-workshop/nginx/access`
2. You should see log streams named with instance IDs
3. Click on a log stream to view logs
4. Verify you can see nginx access logs

✅ **Logs are flowing to CloudWatch!**

Part 7: CloudWatch Logs Insights Queries

Step 1: Navigate to Logs Insights

1. In CloudWatch console
2. Click **Logs** → **Logs Insights** in left navigation menu

Step 2: Basic Query - View All Logs

1. Select log group: `/aws/ec2/sre-workshop/nginx/access`
2. In the query editor, enter:

```sql
fields @timestamp, @message
| sort @timestamp desc
| limit 20
```

3. Click **Run query**
4. View results

Step 3: Query - Count Requests by Status Code

```sql
fields @timestamp, @message
| parse @message /(?<method>\S+)\s+(?<url>\S+)\s+HTTP\/\S+"\s+(?<status>\d+)/
| stats count() by status
| sort status
```

Click **Run query**

Step 4: Query - Top 10 Requested URLs

```sql
fields @timestamp, @message
| parse @message /(?<method>\S+)\s+(?<url>\S+)\s+HTTP/
| stats count() as requests by url
| sort requests desc
| limit 10
```

Click **Run query**

Step 5: Query - Error Rate (4xx and 5xx)

```sql
fields @timestamp, @message
| parse @message /(?<method>\S+)\s+(?<url>\S+)\s+HTTP\/\S+"\s+(?<status>\d+)/
| filter status >= 400
| stats count() as errors by bin(5m)
```

Click **Run query**

Step 6: Query - Request Latency Analysis

First, modify nginx config to log response times:

```bash
sudo nano /etc/nginx/nginx.conf
```

Find the `log_format` section and add (or modify):

```nginx
log_format main '$remote_addr - $remote_user [$time_local] "$request" '
            '$status $body_bytes_sent "$http_referer" '
            '"$http_user_agent" $request_time';
```

Save and restart nginx:

```bash
sudo systemctl restart nginx
```

Generate traffic:

```bash
for i in {1..200}; do curl localhost; done
```

Now run this query:

```sql
fields @timestamp, @message
| parse @message /(?<method>\S+)\s+(?<url>\S+)\s+HTTP\/\S+"\s+(?<status>\d+)\s+\d+\s+"[^"]*"\s+"[^"]*"\s+(?<duration
| stats avg(duration) as avg_latency, max(duration) as max_latency, min(duration) as min_latency by bin(5m)
```

Step 7: Query - Traffic Volume Over Time

```sql
fields @timestamp
| stats count() as requests by bin(1m)
| sort @timestamp desc
```

Click **Run query**

Step 8: Save Query

1. After running a query you want to keep

2. Click **Save** button (top right)
3. Query name: `Request-Count-By-Status`
4. Click **Save**

✅ **Now you can reuse saved queries!**

Part 8: Create Dashboard with Metrics and Logs

Step 1: Navigate to Dashboards

1. Click **Dashboards** in left navigation menu
2. Click **Create dashboard** button
3. Dashboard name: `SRE-Production-Dashboard-Full`
4. Click **Create dashboard**

Step 2: Add Widget - CPU Utilization

1. Select **Line** widget type
2. Click **Next**
3. Click **Metrics** tab
4. Click **EC2 → Per-Instance Metrics**
5. Search for: `CPUUtilization`
6. Check boxes for both instances
7. Click **Create widget** button

Step 3: Add Widget - Memory Usage

⚠️ **IMPORTANT:** If `SREWorkshop` namespace is not visible, wait 2-3 more minutes and refresh!

1. Click **Add widget** (plus icon)
2. Select **Line** widget
3. Click **Next**
4. Click **Browse** tab
5. Select **SREWorkshop** namespace
6. Click **Metrics with no dimensions**
7. Select `mem_used_percent`
8. Click **Options** tab
9. Title: `Memory Utilization %`

10. Left Y-axis: Min `0`, Max `100`
11. Click **Create widget**

Step 4: Add Widget - Disk Usage

1. Click **Add widget**
2. Select **Number** widget type
3. Click **Next**
4. Click **Browse** tab
5. Navigate: **SREWorkshop → Metrics with no dimensions**
6. Select `disk_used_percent`
7. Click **Options** tab
8. Title: `Disk Usage %`
9. Click **Create widget**

Step 5: Add Widget - Network Traffic

1. Click **Add widget**
2. Select **Line** widget
3. Click **Next**
4. Click **Browse → EC2 → Per-Instance Metrics**
5. Search: `NetworkIn`
6. Select both instances
7. Search: `NetworkOut`
8. Select both instances
9. Click **Options** tab
10. Title: `Network Traffic (Bytes)`
11. Click **Create widget**

Step 6: Add Widget - Log Query Results (Request Count)

🆕 **NEW: Adding Log-based Widgets**

1. Click **Add widget**
2. Select **Line** widget
3. Click **Next**
4. Click **Query results** tab (NOT Metrics!)
5. Select **Logs Insights**

6. Log groups: Select `/aws/ec2/sre-workshop/nginx/access`
7. In query editor, enter:

```sql
fields @timestamp
| stats count() as requests by bin(5m)
```

8. Click **Run query**
9. Click **Options** tab
10. Title: `Request Count (5min intervals)`
11. Click **Create widget**

Step 7: Add Widget - Error Rate from Logs

1. Click **Add widget**
2. Select **Line** widget
3. Click **Next**
4. Click **Query results** tab
5. Select **Logs Insights**
6. Log groups: Select `/aws/ec2/sre-workshop/nginx/access`
7. In query editor, enter:

```sql
fields @timestamp, @message
| parse @message /(?<method>\S+)\s+(?<url>\S+)\s+HTTP\/\S+"\s+(?<status>\d+)/
| filter status >= 400
| stats count() as errors by bin(5m)
```

8. Click **Run query**
9. Click **Options** tab
10. Title: `Error Count (4xx/5xx)`
11. Click **Create widget**

Step 8: Add Widget - Top URLs Table

1. Click **Add widget**
2. Select **Table** widget type (or **Bar** for visualization)
3. Click **Next**

4. Click **Query results** tab
5. Select **Logs Insights**
6. Log groups: Select `/aws/ec2/sre-workshop/nginx/access`
7. In query editor, enter:

```sql
fields @timestamp, @message
| parse @message /(?<method>\S+)\s+(?<url>\S+)\s+HTTP/
| stats count() as requests by url
| sort requests desc
| limit 10
```

8. Click **Run query**
9. Click **Options** tab
10. Title: `Top 10 Requested URLs`
11. Click **Create widget**

Step 9: Add Widget - Status Code Distribution

1. Click **Add widget**
2. Select **Pie** widget type
3. Click **Next**
4. Click **Query results** tab
5. Select **Logs Insights**
6. Log groups: Select `/aws/ec2/sre-workshop/nginx/access`
7. In query editor, enter:

```sql
fields @timestamp, @message
| parse @message /(?<method>\S+)\s+(?<url>\S+)\s+HTTP\/\S+"\s+(?<status>\d+)/
| stats count() as requests by status
```

8. Click **Run query**
9. Click **Options** tab
10. Title: `Status Code Distribution`
11. Click **Create widget**

Step 10: Add Widget - Recent Error Logs

1. Click **Add widget**
2. Select **Logs table** widget type
3. Click **Next**
4. Log groups: Select `/aws/ec2/sre-workshop/nginx/error`
5. (Optional) Add filter pattern: `error`
6. Click **Options** tab
7. Title: `Recent Error Logs`
8. Number of rows: `10`
9. Click **Create widget**

Step 11: Arrange Dashboard

Suggested layout:

- **Top row:** Alarm status (if added later) - full width
- **Second row:** CPU and Memory (side by side)
- **Third row:** Network Traffic and Disk Usage (side by side)
- **Fourth row:** Request Count and Error Count from logs (side by side)
- **Fifth row:** Status Code Distribution (Pie) and Top URLs (Table) - side by side
- **Bottom row:** Recent Error Logs - full width

Step 12: Save Dashboard

1. Click **Save dashboard** button (top right)
2. Confirmation message appears

Part 9: Configure Time Range & Auto-Refresh

1. Click time range dropdown (top right)
2. Select **1h** (1 hour)
3. Click **Auto refresh** dropdown
4. Select **1m** (1 minute auto-refresh)

Note: Log widgets may take 1-2 minutes to refresh

Part 10: Create Metric Filters from Logs

Metric filters extract metrics from log data for alarms and tracking

Step 1: Navigate to Log Groups

1. Click **Logs → Log groups**
2. Click on `/aws/ec2/sre-workshop/nginx/access`

Step 2: Create Metric Filter - HTTP 4xx Errors

1. Click **Actions** dropdown
2. Click **Create metric filter**
3. Filter pattern:

```
[ip, id, user, timestamp, request, status_code=4*, size, ...]
```

4. Click **Test pattern**
5. If matches found, click **Next**
6. Filter name: `HTTP-4xx-Errors`
7. Metric namespace: `SREWorkshop/Logs`
8. Metric name: `4xxErrorCount`
9. Metric value: `1`
10. Default value: `0`
11. Click **Next**
12. Review and click **Create metric filter**

Step 3: Create Metric Filter - HTTP 5xx Errors

1. Repeat steps 1-2
2. Filter pattern:

```
[ip, id, user, timestamp, request, status_code=5*, size, ...]
```

3. Continue with:
   - Filter name: `HTTP-5xx-Errors`
   - Metric namespace: `SREWorkshop/Logs`
   - Metric name: `5xxErrorCount`
   - Metric value: `1`
   - Default value: `0`
4. Click **Create metric filter**

Step 4: Create Metric Filter - Request Count

1. Repeat steps 1-2
2. Filter pattern:

```
[ip, id, user, timestamp, request, status_code, size, ...]
```

3. Continue with:
   - Filter name: `Total-Request-Count`
   - Metric namespace: `SREWorkshop/Logs`
   - Metric name: `RequestCount`
   - Metric value: `1`
   - Default value: `0`
4. Click **Create metric filter**

Step 5: Add Metric Filter Metrics to Dashboard

1. Return to dashboard: `SRE-Production-Dashboard-Full`
2. Click **Add widget**
3. Select **Line** widget
4. Click **Next**
5. Click **Browse** tab
6. Select namespace: `SREWorkshop/Logs`
7. Select all three metrics:
   - `4xxErrorCount`
   - `5xxErrorCount`
   - `RequestCount`
8. Click **Options** tab
9. Title: `Log-Based Metrics`
10. Click **Create widget**
11. Click **Save dashboard**

✅ **Now you have metrics extracted from logs!**

Part 11: Create CloudWatch Alarms

Step 1: Navigate to Alarms

1. In CloudWatch console

2. Click **Alarms** in left menu
3. Click **All alarms**
4. Click **Create alarm** button

Step 2: Create CPU Alarm

1. Click **Select metric**
2. Navigate: **EC2 → Per-Instance Metrics**
3. Search: `CPUUtilization`
4. Select one instance
5. Click **Select metric**

Step 3: Configure Alarm Conditions

1. Statistic: `Average`
2. Period: `5 minutes`
3. Threshold type: `Static`
4. Condition: `Greater than 80`
5. Click **Next**

Step 4: Configure Notifications

1. Select: **Create new topic**
2. Topic name: `SRE-Alerts`
3. Email: Enter your email address
4. Click **Create topic**
5. Click **Next**

Step 5: Name Alarm

1. Alarm name: `High-CPU-Web-01`
2. Description: `CPU exceeds 80% threshold`
3. Click **Next**
4. Review settings
5. Click **Create alarm**

Step 6: Confirm Email Subscription

1. Check your email
2. Click confirmation link in AWS notification email

3. Return to CloudWatch console

Step 7: Create Additional Alarms

Create alarms for:

- **High Memory Usage:** `mem_used_percent > 85`
- **Disk Space Low:** `disk_used_percent > 90`
- **High 4xx Error Rate:** `4xxErrorCount > 10` (from log metric filter)
- **High 5xx Error Rate:** `5xxErrorCount > 5` (from log metric filter)
- **Status Check Failed:** `StatusCheckFailed > 0`

Part 12: Create Log-Based Alarm

Step 1: Create Alarm from Logs Insights

1. Go to **Logs Insights**
2. Run a query, for example:

```sql
fields @timestamp, @message
| parse @message /(?<method>\S+)\s+(?<url>\S+)\s+HTTP\/\S+"\s+(?<status>\d+)/
| filter status >= 500
| stats count() as errors by bin(5m)
```

3. After running query, click **Actions**
4. Click **Create metric filter**
5. Continue creating metric filter as in Part 10
6. Then create alarm on that metric

Part 13: Add Alarms to Dashboard

Step 1: Open Dashboard

1. Click **Dashboards** in left menu
2. Click **SRE-Production-Dashboard-Full**

Step 2: Add Alarm Widget

1. Click **Add widget**

2. Select **Alarm status** widget
3. Click **Next**
4. Click **Configure**
5. Select all alarms created
6. Click **Create widget**

Step 3: Position Widget

1. Drag alarm widget to top of dashboard
2. Resize to full width
3. Click **Save dashboard**

Part 14: Create SLO Tracking

Step 1: Calculate Availability SLO

1. Navigate to **Metrics → All metrics**
2. Click **Browse** tab
3. Select **EC2 → Per-Instance Metrics**
4. Select: `StatusCheckFailed`

Step 2: Create Math Expression

1. Click **Add math → Start with empty expression**
2. Enter formula: `100 - (m1 * 100)`
3. Label: `Availability %`
4. ID: `e1`

Step 3: Add to Dashboard

1. Click **Actions → Add to dashboard**
2. Select: `SRE-Production-Dashboard-Full`
3. Widget type: **Number**
4. Click **Add to dashboard**
5. Click **Save dashboard**

Part 15: Advanced Log Analysis

Query 1: Find Slow Requests (if response time is logged)

```sql
fields @timestamp, @message
| parse @message /(?<method>\S+)\s+(?<url>\S+)\s+HTTP\/\S+"\s+(?<status>\d+)\s+\d+\s+"[^"]*"\s+"[^"]*"\s+(?<duration
| filter duration > 1.0
| sort duration desc
| limit 20
```

Query 2: Traffic by Hour

```sql
fields @timestamp
| stats count() as requests by bin(1h)
| sort @timestamp desc
```

Query 3: Unique IP Addresses

```sql
fields @timestamp, @message
| parse @message /^(?<ip>\S+)/
| stats count_distinct(ip) as unique_ips by bin(1h)
```

Query 4: User Agent Analysis

```sql
fields @timestamp, @message
| parse @message /"(?<user_agent>[^"]*)"$/
| stats count() as requests by user_agent
| sort requests desc
| limit 10
```

Query 5: Error Log Pattern Detection

For error logs (`/aws/ec2/sre-workshop/nginx/error`):

```sql
fields @timestamp, @message
| filter @message like /error/
| stats count() as error_count by @message
| sort error_count desc
| limit 10
```

Part 16: Test & Validate

Step 1: Generate Load

Return to EC2 instance terminal:

```bash
sudo yum install stress -y
stress --cpu 2 --timeout 300s
```

Step 2: Generate Web Traffic with Errors

Create a test script:

```bash
cat > /tmp/generate_traffic.sh << 'EOF'
#!/bin/bash
for i in {1..1000}; do
  # Normal requests
  curl -s http://localhost/ > /dev/null

  # 404 errors
  curl -s http://localhost/nonexistent > /dev/null

  # Random delay
  sleep 0.1
done
EOF

chmod +x /tmp/generate_traffic.sh
/tmp/generate_traffic.sh &
```

Step 3: Monitor Dashboard

1. Return to CloudWatch dashboard
2. Watch metrics update in real-time
3. Observe CPU spike in metric widgets
4. Observe request counts in log widgets
5. Check error rate widgets
6. Wait for alarm notification email

Step 4: Check Logs Insights

1. Go to **Logs Insights**
2. Run queries to analyze the generated traffic
3. Look for patterns and anomalies

Step 5: Stop Load Test

In terminal:

```bash
# Stop CPU stress
# Press Ctrl+C if still running

# Stop traffic generation
pkill -f generate_traffic
```

Troubleshooting Guide

Issue: Log Groups Not Appearing

Solutions:

1. **Check IAM Role:**

```bash
curl http://169.254.169.254/latest/meta-data/iam/security-credentials/
```

- Verify it includes CloudWatch Logs permissions

2. **Verify Agent Status:**

```bash
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -m ec2 -a status -c default
```

3. **Check Agent Logs:**

```bash
sudo tail -f /opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-agent.log
```

4. **Verify Log Files Exist:**

```bash
ls -la /var/log/nginx/
sudo tail -f /var/log/nginx/access.log
```

Issue: SREWorkshop Namespace Not Appearing

Solutions:

1. **Check IAM Role attached to EC2**
2. **Verify agent is running**
3. **Wait 3-5 minutes for first metrics**
4. **Restart agent if needed:**

```bash
sudo systemctl restart amazon-cloudwatch-agent
```

Issue: Log Insights Query Not Returning Results

Solutions:

1. **Check time range** - expand to cover when logs were generated
2. **Verify log group has data** - check log streams
3. **Test parse pattern** - simplify query to just view raw logs first
4. **Check filter syntax** - ensure proper CloudWatch Logs Insights syntax

---

Workshop Summary & Best Practices

Key Takeaways

Dashboard Organization:

- Combine metrics and logs for complete observability
- Use log-based widgets for application-level insights
- Group related metrics and logs together
- Use consistent time ranges
- Enable auto-refresh for production monitoring

Log Analysis Best Practices:

- Use structured logging for easier parsing

- Create metric filters for important patterns
- Save frequently-used queries
- Set up alerts on log-based metrics
- Use appropriate retention periods

Alarm Strategy:

- Set thresholds based on historical data
- Create alarms on both metrics and log patterns
- Use multiple evaluation periods to avoid false positives
- Create escalation paths (warning → critical)
- Test alarms regularly

SRE Metrics Priority:

1. **Availability** (uptime)
2. **Latency** (response time from logs)
3. **Error rate** (from both metrics and logs)
4. **Saturation** (resource usage)

Next Steps

1. Advanced Log Analysis:

- Implement distributed tracing
- Correlate metrics with log events
- Create custom log parsing rules
- Set up anomaly detection

2. Enhanced Dashboards:

- Create executive summary dashboard
- Build team-specific dashboards
- Add business metrics from logs
- Implement custom visualizations

3. Automation:

- Auto-scaling based on log patterns
- Automated log-based remediation

- Incident response workflows
- Log archiving to S3

4. Advanced Monitoring:

- Application Performance Monitoring (APM)
- Real User Monitoring (RUM)
- Synthetic monitoring
- Container and serverless monitoring

Cleanup Instructions

1. Delete Alarms:

CloudWatch → Alarms → Select all → Actions → Delete

2. Delete Metric Filters:

CloudWatch → Logs → Log groups → Select group → Metric filters → Delete

3. Delete Dashboard:

CloudWatch → Dashboards → Select → Delete

4. Delete Log Groups:

CloudWatch → Logs → Log groups → Select all → Actions → Delete log group

5. Terminate EC2 Instances:

EC2 → Instances → Select → Instance state → Terminate

6. Delete SNS Topics:

SNS → Topics → Select → Delete

7. Delete IAM Role:

IAM → Roles → Select `SRE-Workshop-CloudWatch-Logs-Role` → Delete

Additional Resources

AWS Documentation:

- CloudWatch User Guide: https://docs.aws.amazon.com/cloudwatch/

- CloudWatch Logs Insights: https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/AnalyzingLogData.html
- EC2 Monitoring Guide: https://docs.aws.amazon.com/ec2/monitoring.html
- IAM Roles for EC2: https://docs.aws.amazon.com/AWSEC2