AWS Security Monitoring Dashboard - Management Console Implementation Guide

Prerequisites and Initial Setup

What You'll Need

- AWS Free Tier account (active and verified)
- Valid email address for notifications
- Modern web browser (Chrome, Firefox, Safari, or Edge)
- Basic understanding of AWS services

Initial Account Setup

1. **Sign in to AWS Management Console**
    - Go to https://aws.amazon.com/console/
    - Click "Sign In to the Console"
    - Enter your account credentials
    - Select your preferred region (recommended: us-east-1 for beginners)
2. **Verify Free Tier Status**
    - In the top navigation bar, click on your account name
    - Select "Billing and Cost Management"
    - Click "Free tier" in the left sidebar
    - Confirm you have available free tier resources

Phase 1: CloudTrail Setup for Security Logging

1.1 Create S3 Bucket for Log Storage

Navigation Path: S3 Console → Create Bucket

1. **Access S3 Console**
    - In the AWS Console search bar, type "S3"
    - Click "S3" from the search results
    - Click "Create bucket" (orange button)
2. **Configure Bucket Settings**
    - **Bucket name**: Enter `security-monitoring-logs-[your-initials]-[random-number]`
        - Example: `security-monitoring-logs-js-2024`

    - Note: Bucket names must be globally unique
  - **AWS Region**: Select your preferred region (keep consistent throughout project)
  - **Object Ownership**: Leave as "ACLs disabled (recommended)"

3. **Configure Block Public Access**
   - Leave all four checkboxes **checked** (this is secure default)
   - Block all public access: ✅ Enabled
4. **Bucket Versioning**
   - Under "Bucket Versioning"
   - Select "Enable" (helps with log integrity)
5. **Default Encryption**
   - Under "Default encryption"
   - Select "Server-side encryption with Amazon S3 managed keys (SSE-S3)"
   - Leave other settings as default
6. **Create Bucket**
   - Click "Create bucket" at the bottom
   - **Verification**: You should see "Successfully created bucket" message

1.2 Configure CloudTrail

Navigation Path: CloudTrail Console → Create Trail

1. **Access CloudTrail Console**
   - In AWS Console search bar, type "CloudTrail"
   - Click "CloudTrail" from search results
   - Click "Create trail" (if this is your first trail)
2. **Trail Configuration**
   - **Trail name**: `security-monitoring-trail`
   - **Apply trail to all regions**: ✅ **Enable this** (important for comprehensive monitoring)
   - **Enable log file validation**: ✅ **Enable this** (detects tampering)
3. **CloudWatch Logs Configuration**
   - **CloudWatch Logs**: ✅ **Enable**
   - **Log group name**: `CloudTrail/SecurityMonitoring`
   - **IAM Role**: Select "Create a new role"
   - **Role name**: `CloudTrail-CloudWatchLogs-Role`
4. **S3 Bucket Configuration**

- **Create a new S3 bucket**: ❌ **Uncheck** (we'll use existing)
- **S3 bucket**: Select the bucket you created in step 1.1
- **Log file prefix**: `AWSLogs/`
- **Encrypt log files with SSE-S3**: ✅ **Enable**

5. **Event Type Selection**
   - **Management events**: ✅ **Enable**
   - **Read**: ✅ **Enable**
   - **Write**: ✅ **Enable**
   - **Data events**: ❌ **Leave disabled** (to stay within free tier)
   - **Insight events**: ❌ **Leave disabled** (to stay within free tier)
6. **Create Trail**
   - Review all settings
   - Click "Create trail"
   - **Verification**: You should see "Trail created successfully" message

---

Phase 2: SNS Setup for Alert Notifications

2.1 Create SNS Topics

Navigation Path: SNS Console → Create Topic

1. **Access SNS Console**
   - Search for "SNS" in AWS Console
   - Click "Simple Notification Service"
   - Click "Create topic"
2. **Create Critical Alerts Topic**
   - **Type**: Select "Standard"
   - **Name**: `security-critical-alerts`
   - **Display name**: `Security Critical Alerts`
   - **Description**: `Critical security events requiring immediate attention`
   - Leave other settings as default
   - Click "Create topic"
   - **Note**: Copy the Topic ARN displayed (you'll need it later)
3. **Create Warning Alerts Topic**
   - Click "Create topic" again

- **Type**: Select "Standard"
- **Name**: `security-warning-alerts`
- **Display name**: `Security Warning Alerts`
- **Description**: `Security warnings requiring attention`
- Click "Create topic"
- **Note**: Copy this Topic ARN as well

2.2 Subscribe to Email Notifications

Navigation Path: SNS Console → Topics → Create Subscription

1. **Subscribe to Critical Alerts**
   - In SNS Console, click "Topics" in left sidebar
   - Click on `security-critical-alerts` topic
   - Click "Create subscription"
   - **Protocol**: Select "Email"
   - **Endpoint**: Enter your email address
   - Click "Create subscription"
2. **Subscribe to Warning Alerts**
   - Go back to Topics list
   - Click on `security-warning-alerts` topic
   - Click "Create subscription"
   - **Protocol**: Select "Email"
   - **Endpoint**: Enter your email address
   - Click "Create subscription"
3. **Confirm Email Subscriptions**
   - Check your email inbox
   - You should receive 2 confirmation emails
   - Click "Confirm subscription" in each email
   - **Verification**: Return to SNS console and verify subscription status shows "Confirmed"

Phase 3: CloudWatch Monitoring Setup

3.1 Create Metric Filters

Navigation Path: CloudWatch Console → Logs → Log Groups → Metric Filters

1. **Access CloudWatch Logs**
   - Search for "CloudWatch" in AWS Console
   - Click "CloudWatch"
   - In left sidebar, click "Logs" → "Log groups"
   - Find and click on `CloudTrail/SecurityMonitoring`
2. **Create Root Account Usage Filter**
   - Click "Metric filters" tab
   - Click "Create metric filter"
   - **Filter pattern**:

     ```
     { ($.userIdentity.type = "Root") && ($.userIdentity.invokedBy NOT EXISTS) &&
     ($.eventType != "AwsServiceEvent") }
     ```

   - Click "Test pattern" to verify
   - Click "Next"
   - **Filter name**: `RootAccountUsage`
   - **Metric namespace**: `SecurityMonitoring`
   - **Metric name**: `RootAccountUsageCount`
   - **Metric value**: `1`
   - **Default value**: `0`
   - Click "Next" → "Create metric filter"
3. **Create Failed Console Login Filter**
   - Click "Create metric filter" again
   - **Filter pattern**:

     ```
     { ($.eventName = ConsoleLogin) && ($.errorMessage EXISTS) }
     ```

   - Click "Test pattern" → "Next"
   - **Filter name**: `FailedConsoleLogins`
   - **Metric namespace**: `SecurityMonitoring`
   - **Metric name**: `FailedConsoleLoginCount`
   - **Metric value**: `1`
   - **Default value**: `0`
   - Click "Next" → "Create metric filter"
4. **Create Unauthorized API Calls Filter**
   - Click "Create metric filter"

- **Filter pattern**:

  ```
  { ($.errorCode = "*UnauthorizedOperation") || ($.errorCode = "AccessDenied*") }
  ```

- Click "Test pattern" → "Next"
- **Filter name**: `UnauthorizedAPICalls`
- **Metric namespace**: `SecurityMonitoring`
- **Metric name**: `UnauthorizedAPICallCount`
- **Metric value**: `1`
- **Default value**: `0`
- Click "Next" → "Create metric filter"

5. **Create Security Group Changes Filter**
   - Click "Create metric filter"
   - **Filter pattern**:

     ```
     { ($.eventName = AuthorizeSecurityGroupIngress) || ($.eventName =
     AuthorizeSecurityGroupEgress) || ($.eventName = RevokeSecurityGroupIngress) ||
     ($.eventName = RevokeSecurityGroupEgress) || ($.eventName =
     CreateSecurityGroup) || ($.eventName = DeleteSecurityGroup) }
     ```

   - Click "Test pattern" → "Next"
   - **Filter name**: `SecurityGroupChanges`
   - **Metric namespace**: `SecurityMonitoring`
   - **Metric name**: `SecurityGroupChangeCount`
   - **Metric value**: `1`
   - **Default value**: `0`
   - Click "Next" → "Create metric filter"
6. **Create IAM Policy Changes Filter**
   - Click "Create metric filter"
   - **Filter pattern**:

     ```
     { ($.eventName=DeleteGroupPolicy) || ($.eventName=DeleteRolePolicy) ||
     ($.eventName=DeleteUserPolicy) || ($.eventName=PutGroupPolicy) ||
     ($.eventName=PutRolePolicy) || ($.eventName=PutUserPolicy) ||
     ($.eventName=CreatePolicy) || ($.eventName=DeletePolicy) ||
     ($.eventName=CreatePolicyVersion) || ($.eventName=DeletePolicyVersion) ||
     ($.eventName=AttachRolePolicy) || ($.eventName=DetachRolePolicy) ||
     ($.eventName=AttachUserPolicy) || ($.eventName=DetachUserPolicy) ||
     ($.eventName=AttachGroupPolicy) || ($.eventName=DetachGroupPolicy) }
     ```

- Click "Test pattern" → "Next"
- **Filter name**: `IAMPolicyChanges`
- **Metric namespace**: `SecurityMonitoring`
- **Metric name**: `IAMPolicyChangeCount`
- **Metric value**: `1`
- **Default value**: `0`
- Click "Next" → "Create metric filter"

3.2 Create CloudWatch Alarms

Navigation Path: CloudWatch Console → Alarms → Create Alarm

1. **Access CloudWatch Alarms**
   - In CloudWatch console, click "Alarms" in left sidebar
   - Click "Create alarm"
2. **Create Root Account Usage Alarm**
   - **Select metric**: Click "Select metric"
   - Browse: `SecurityMonitoring` → Select `RootAccountUsageCount`
   - Click "Select metric"
   - **Statistic**: Sum
   - **Period**: 5 minutes
   - **Threshold type**: Static
   - **Comparison**: Greater/Equal
   - **Threshold value**: `1`
   - Click "Next"
   - **Alarm state trigger**: In alarm
   - **SNS topic**: Select `security-critical-alerts`
   - Click "Next"
   - **Alarm name**: `Root Account Usage Alert`
   - **Alarm description**: `Alert when root account is used`
   - Click "Next" → "Create alarm"
3. **Create Failed Login Alarm**
   - Click "Create alarm"
   - **Select metric**: `SecurityMonitoring` → `FailedConsoleLoginCount`

- **Statistic**: Sum
- **Period**: 5 minutes
- **Threshold**: Greater/Equal to `3`
- **SNS topic**: Select `security-warning-alerts`
- **Alarm name**: `Multiple Failed Console Logins`
- **Description**: `Alert when multiple failed console logins occur`
- Create alarm

4. **Create Unauthorized API Calls Alarm**
   - Click "Create alarm"
   - **Select metric**: `SecurityMonitoring` → `UnauthorizedAPICallCount`
   - **Statistic**: Sum
   - **Period**: 5 minutes
   - **Threshold**: Greater/Equal to `10`
   - **SNS topic**: Select `security-warning-alerts`
   - **Alarm name**: `High Unauthorized API Calls`
   - **Description**: `Alert when unauthorized API calls are detected`
   - Create alarm
5. **Create Security Group Changes Alarm**
   - Click "Create alarm"
   - **Select metric**: `SecurityMonitoring` → `SecurityGroupChangeCount`
   - **Statistic**: Sum
   - **Period**: 5 minutes
   - **Threshold**: Greater/Equal to `1`
   - **SNS topic**: Select `security-warning-alerts`
   - **Alarm name**: `Security Group Changes`
   - **Description**: `Alert when security groups are modified`
   - Create alarm
6. **Create IAM Policy Changes Alarm**
   - Click "Create alarm"
   - **Select metric**: `SecurityMonitoring` → `IAMPolicyChangeCount`
   - **Statistic**: Sum
   - **Period**: 5 minutes

- **Threshold**: Greater/Equal to `1`
- **SNS topic**: Select `security-critical-alerts`
- **Alarm name**: `IAM Policy Changes`
- **Description**: `Alert when IAM policies are modified`
- Create alarm

Verification: You should now see 5 alarms in your CloudWatch Alarms dashboard

---

Phase 4: Enhanced Processing with Lambda

4.1 Create IAM Role for Lambda

Navigation Path: IAM Console → Roles → Create Role

1. **Access IAM Console**
   - Search for "IAM" in AWS Console
   - Click "IAM"
   - Click "Roles" in left sidebar
   - Click "Create role"
2. **Configure Role**
   - **Trusted entity type**: AWS service
   - **Use case**: Lambda
   - Click "Next"
3. **Add Permissions**
   - Search for and select these policies:
     - `AWSLambdaBasicExecutionRole` ✅
   - Click "Next"
4. **Role Details**
   - **Role name**: `SecurityMonitoringLambdaRole`
   - **Description**: `Role for security monitoring Lambda functions`
   - Click "Create role"
5. **Add Custom Permissions**
   - Click on the role you just created
   - Click "Add permissions" → "Create inline policy"
   - Click "JSON" tab
   - Paste this policy:

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sns:Publish",
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "cloudwatch:PutMetricData",
        "s3:GetObject"
      ],
      "Resource": "*"
    }
  ]
}
```

- Click "Next: Tags" → "Next: Review"
- **Name**: `SecurityMonitoringCustomPolicy`
- Click "Create policy"

4.2 Create Lambda Function

Navigation Path: Lambda Console → Create Function

1. **Access Lambda Console**
   - Search for "Lambda" in AWS Console
   - Click "Lambda"
   - Click "Create function"
2. **Function Configuration**
   - **Function option**: Author from scratch
   - **Function name**: `SecurityAlertProcessor`
   - **Runtime**: Python 3.9
   - **Architecture**: x86_64
   - **Execution role**: Use an existing role
   - **Existing role**: Select `SecurityMonitoringLambdaRole`
   - Click "Create function"
3. **Function Code**

- In the code editor, replace the default code with:

python

```python
import json
import boto3
from datetime import datetime

def lambda_handler(event, context):
    sns = boto3.client('sns')

    try:
        # Parse CloudWatch alarm from SNS
        message = json.loads(event['Records'][0]['Sns']['Message'])
        alarm_name = message['AlarmName']
        new_state = message['NewStateValue']
        reason = message['NewStateReason']
        timestamp = message['StateChangeTime']

        # Determine severity
        critical_alarms = ['Root Account Usage Alert', 'IAM Policy Changes']
        severity = "🚨 CRITICAL" if alarm_name in critical_alarms else "⚠️ WARNING"

        # Create enhanced alert message
        alert_message = f"""
{severity} SECURITY ALERT

Alarm: {alarm_name}
State: {new_state}
Time: {timestamp}
Reason: {reason}

Immediate Actions Required:
1. Review CloudTrail logs for this time period
2. Verify if this activity was authorized
3. Check for additional suspicious activities
4. Document incident response actions taken

AWS Console: https://console.aws.amazon.com/cloudwatch/home#alarms:
        """

        # Get the original SNS topic ARN from the event
        topic_arn = event['Records'][0]['Sns']['TopicArn']

        # Send enhanced notification back to the same topic
        response = sns.publish(
            TopicArn=topic_arn,
            Message=alert_message,
            Subject=f'{severity.split()[0]} Security Alert: {alarm_name}'
        )
```

```
            return {
                'statusCode': 200,
                'body': json.dumps('Alert processed and enhanced successfully')
            }

        except Exception as e:
            print(f'Error processing alert: {str(e)}')
            return {
                'statusCode': 500,
                'body': json.dumps(f'Error: {str(e)}')
            }
```

- Click "Deploy" (orange button)

4. **Add SNS Triggers**
   - Click "Add trigger"
   - **Trigger configuration**: SNS
   - **SNS topic**: Select `security-critical-alerts`
   - ✅ **Enable trigger**
   - Click "Add"
   - Click "Add trigger" again
   - **Trigger configuration**: SNS
   - **SNS topic**: Select `security-warning-alerts`
   - ✅ **Enable trigger**
   - Click "Add"

Verification: Your Lambda function should show 2 SNS triggers in the function overview

---

Phase 5: Create Security Dashboard

5.1 Build CloudWatch Dashboard

Navigation Path: CloudWatch Console → Dashboards → Create Dashboard

1. **Access CloudWatch Dashboards**
   - In CloudWatch console, click "Dashboards" in left sidebar
   - Click "Create dashboard"
2. **Dashboard Configuration**
   - **Dashboard name**: `SecurityMonitoring`
   - Click "Create dashboard"

3. **Add Security Events Timeline Widget**
   - **Widget type**: Line graph
   - Click "Configure"
   - **Metrics**: Click "Add metric"
   - **Browse**: `SecurityMonitoring`
   - Select all 5 metrics:
     - `RootAccountUsageCount`
     - `FailedConsoleLoginCount`
     - `UnauthorizedAPICallCount`
     - `SecurityGroupChangeCount`
     - `IAMPolicyChangeCount`
   - **Statistic**: Sum
   - **Period**: 5 minutes
   - **Widget title**: `Security Events Timeline`
   - Click "Create widget"
4. **Add CloudTrail Activity Widget**
   - Click "Add widget"
   - **Widget type**: Line graph
   - **Metrics**: Browse to `AWS/CloudTrail`
   - Add available CloudTrail metrics
   - **Widget title**: `CloudTrail Activity`
   - Click "Create widget"
5. **Add Single Value Metrics**
   - Click "Add widget"
   - **Widget type**: Number
   - **Metrics**: `SecurityMonitoring` → `RootAccountUsageCount`
   - **Statistic**: Sum
   - **Period**: 1 day
   - **Widget title**: `Root Account Usage (24h)`
   - Click "Create widget"
6. **Add Log Insights Widget**
   - Click "Add widget"
   - **Widget type**: Logs table

- **Log groups**: Select `CloudTrail/SecurityMonitoring`
- **Query**:

```sql
fields @timestamp, sourceIPAddress, userIdentity.type, eventName, errorCode
| filter errorCode exists
| sort @timestamp desc
| limit 20
```

- **Widget title**: `Recent Failed API Calls`
- Click "Create widget"

7. **Save Dashboard**
   - Click "Save dashboard" at the top
   - Your dashboard is now created and will refresh automatically

---

Phase 6: Testing and Verification

6.1 Generate Test Security Events

Navigation Path: EC2 Console → Security Groups

1. **Create Test Security Group**
   - Search for "EC2" in AWS Console
   - Click "EC2"
   - In left sidebar, click "Security Groups"
   - Click "Create security group"
   - **Name**: `test-monitoring-sg`
   - **Description**: `Test security group for monitoring alerts`
   - **VPC**: Select default VPC
   - Click "Create security group"
2. **Modify Security Group (Triggers Alert)**
   - Click on the security group you just created
   - Click "Edit inbound rules"
   - Click "Add rule"
   - **Type**: HTTP
   - **Source**: Anywhere-IPv4
   - Click "Save rules"

- **Expected Result**: This should trigger the Security Group Changes alarm

3. **Clean Up Test Resources**
   - Select the test security group
   - Click "Actions" → "Delete security group"
   - Confirm deletion

6.2 Verify Monitoring System

Verification Checklist:

1. **Check CloudTrail Logging**
   - Go to CloudTrail console
   - Click on your trail name
   - Click "Event history"
   - Verify recent events are being logged
   - ✅ **Expected**: You should see CreateSecurityGroup and DeleteSecurityGroup events
2. **Verify Metric Filters**
   - Go to CloudWatch → Logs → Log groups
   - Click on `CloudTrail/SecurityMonitoring`
   - Click "Metric filters" tab
   - ✅ **Expected**: You should see 5 metric filters
3. **Check Alarm Status**
   - Go to CloudWatch → Alarms
   - ✅ **Expected**: You should see 5 alarms, with "Security Group Changes" possibly in ALARM state
4. **Verify Email Notifications**
   - Check your email inbox
   - ✅ **Expected**: You should receive email notification about security group changes
5. **Test Dashboard**
   - Go to CloudWatch → Dashboards
   - Click on "SecurityMonitoring"
   - ✅ **Expected**: Dashboard loads with widgets showing data
6. **Verify Lambda Function**
   - Go to Lambda console
   - Click on "SecurityAlertProcessor"
   - Click "Monitor" tab

- ✅ **Expected**: You should see recent invocations if alarms triggered

---

Ongoing Monitoring and Maintenance

Daily Monitoring Tasks

1. **Check Dashboard**: Review SecurityMonitoring dashboard for unusual activity
2. **Review Alerts**: Investigate any security alerts received via email
3. **Verify Service Health**: Ensure all alarms are in OK state (unless there's a real issue)

Weekly Maintenance

1. **Review Metrics**: Check if alarm thresholds need adjustment
2. **Log Analysis**: Review CloudTrail logs for patterns
3. **Cost Monitoring**: Verify staying within free tier limits

Monthly Reviews

1. **Alarm Effectiveness**: Analyze false positive rates
2. **Security Posture**: Review overall security metrics trends
3. **Documentation**: Update any procedures or contact information

---

Troubleshooting Common Issues

Issue: Not Receiving Email Alerts

Solution:

1. Go to SNS Console → Topics
2. Click on your topic → Subscriptions
3. Verify subscription status is "Confirmed"
4. Check spam folder
5. Re-subscribe if needed

Issue: Alarms Not Triggering

Solution:

1. Go to CloudWatch → Logs → Log groups
2. Verify CloudTrail logs are being received
3. Check metric filters are correctly configured
4. Test metric filters with sample data

Issue: Lambda Function Errors

Solution:

1. Go to Lambda Console → Functions → SecurityAlertProcessor
2. Click "Monitor" tab → "View logs in CloudWatch"
3. Review error logs for debugging
4. Check IAM permissions are correct

Issue: High AWS Costs

Solution:

1. Go to Billing Console → Cost Explorer
2. Identify which services are generating costs
3. Adjust log retention periods
4. Review and optimize resource usage

Free Tier Optimization Tips

Staying Within Limits

- **CloudWatch Logs**: Set retention to 7-14 days max
- **Lambda**: Monitor execution time and memory usage
- **SNS**: Limit to essential notifications only
- **S3**: Use lifecycle policies to transition old logs to cheaper storage

Cost Monitoring Setup

1. Go to Billing Console
2. Click "Budgets" → "Create budget"
3. Set alerts at $1, $5, and $10 spending levels
4. Configure email notifications for budget alerts

Your AWS Security Monitoring Dashboard is now fully operational! You have comprehensive security monitoring that will alert you to suspicious activities in real-time while staying within AWS free tier limits.