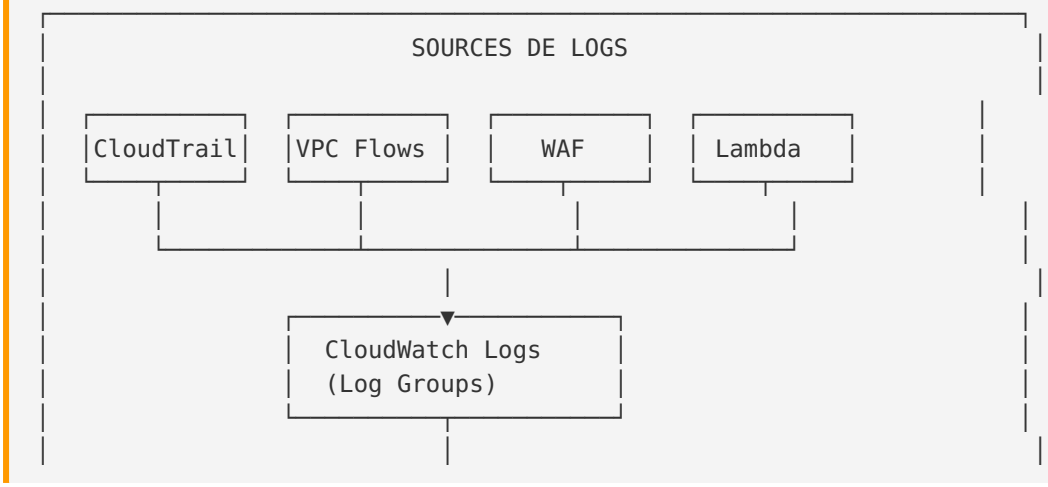
Guide Complet : Supervision Sécurité avec AWS CloudWatch

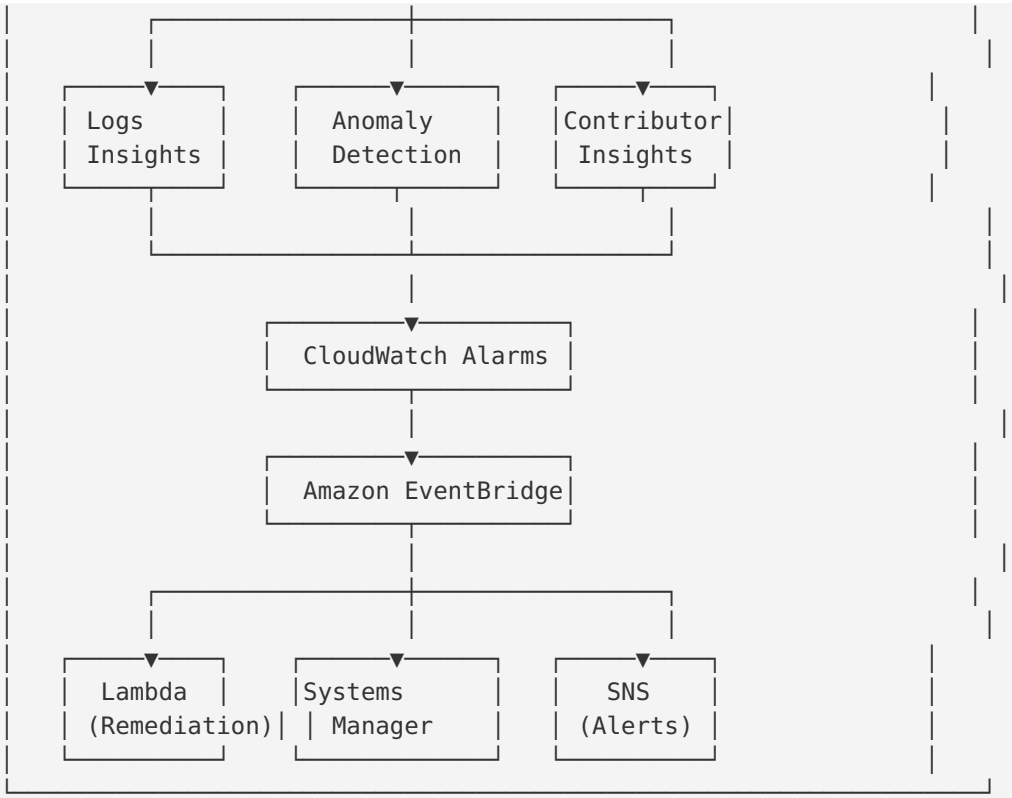
Version: 1.0
Date: Novembre 2025
Destiné à: Équipes SRE et Sécurité

Table des Matières

Architecture de Supervision Sécurité

1. Vue d'Ensemble





2. Stratégie de Logs Centralisés

Bonnes pratiques 2025:

- ✅ **Centralisation** : Tous les logs dans un compte dédié
- ✅ **Rétention** : Minimum 90 jours (compliance), 1-2 ans (forensics)
- ✅ **Chiffrement** : KMS pour tous les log groups
- ✅ **Automatisation** : Alertes et réponses automatisées

```
# Créer un log group centralisé avec chiffrement
aws logs create-log-group --log-group-name /security/centralized-logs

aws logs put-retention-policy \
    --log-group-name /security/centralized-logs \
    --retention-in-days 365

# Activer le chiffrement KMS
aws logs associate-kms-key \
    --log-group-name /security/centralized-logs \
    --kms-key-id arn:aws:kms:us-east-1:123456789012:key/xxxxx
```

CloudWatch Alarmes Critiques

1. Alarmes IAM et Accès

1.1 Changements de Politiques IAM

```
# Créer un filtre de métrique pour les changements de politique
aws logs put-metric-filter \
    --log-group-name /aws/cloudtrail/logs \
    --filter-name IAM-Policy-Changes \
    --filter-pattern '{($.eventName=DeleteGroupPolicy) || ($.eventName=DeleteRolePolicy) || ($
    --metric-transformations \
        metricName=IAMPolicyChanges,metricNamespace=CloudTrailMetrics,metricValue=1

# Créer l'alarme
aws cloudwatch put-metric-alarm \
    --alarm-name IAM-Policy-Change-Detected \
    --alarm-description "Alerte lors d'un changement de politique IAM" \
    --metric-name IAMPolicyChanges \
    --namespace CloudTrailMetrics \
    --statistic Sum \
    --period 300 \
    --evaluation-periods 1 \
    --threshold 1 \
    --comparison-operator GreaterThanOrEqualToThreshold \
    --treat-missing-data notBreaching \
    --alarm-actions arn:aws:sns:us-east-1:123456789012:SecurityAlerts
```

1.2 Utilisation du Compte Root

```
# Filtre pour détecter l'utilisation du compte root
aws logs put-metric-filter \
    --log-group-name /aws/cloudtrail/logs \
    --filter-name Root-Account-Usage \
    --filter-pattern '{$.userIdentity.type="Root" && $.userIdentity.invokedBy NOT EXISTS && $.
    --metric-transformations \
        metricName=RootAccountUsage,metricNamespace=CloudTrailMetrics,metricValue=1

aws cloudwatch put-metric-alarm \
    --alarm-name Root-Account-Usage-Detected \
    --alarm-description "CRITICAL: Root account was used" \
    --metric-name RootAccountUsage \
    --namespace CloudTrailMetrics \
    --statistic Sum \
    --period 60 \
    --evaluation-periods 1 \
    --threshold 1 \
    --comparison-operator GreaterThanOrEqualToThreshold \
    --alarm-actions arn:aws:sns:us-east-1:123456789012:CriticalSecurityAlerts
```

1.3 Échecs de Connexion Console

```
# Détecter les tentatives de connexion échouées
aws logs put-metric-filter \
    --log-group-name /aws/cloudtrail/logs \
    --filter-name Console-Login-Failures \
    --filter-pattern '{($.eventName=ConsoleLogin) && ($.errorMessage="Failed authentication")}
    --metric-transformations \
        metricName=ConsoleLoginFailures,metricNamespace=CloudTrailMetrics,metricValue=1

aws cloudwatch put-metric-alarm \
    --alarm-name Excessive-Console-Login-Failures \
    --alarm-description "Multiple failed console login attempts detected" \
    --metric-name ConsoleLoginFailures \
    --namespace CloudTrailMetrics \
    --statistic Sum \
    --period 300 \
    --evaluation-periods 1 \
    --threshold 5 \
    --comparison-operator GreaterThanThreshold \
    --alarm-actions arn:aws:sns:us-east-1:123456789012:SecurityAlerts
```

2. Alarmes Infrastructure

2.1 Changements de Security Groups

```
# Détecter les modifications de security groups
aws logs put-metric-filter \
    --log-group-name /aws/cloudtrail/logs \
    --filter-name Security-Group-Changes \
    --filter-pattern '{($.eventName=AuthorizeSecurityGroupIngress) || ($.eventName=AuthorizeSe
    --metric-transformations \
        metricName=SecurityGroupChanges,metricNamespace=CloudTrailMetrics,metricValue=1

aws cloudwatch put-metric-alarm \
    --alarm-name Security-Group-Change-Detected \
    --metric-name SecurityGroupChanges \
    --namespace CloudTrailMetrics \
    --statistic Sum \
    --period 300 \
    --evaluation-periods 1 \
    --threshold 1 \
    --comparison-operator GreaterThanOrEqualToThreshold \
    --alarm-actions arn:aws:sns:us-east-1:123456789012:InfrastructureAlerts
```

2.2 Clés KMS Désactivées

```
# Alerter si une clé KMS est désactivée ou supprimée
aws logs put-metric-filter \
    --log-group-name /aws/cloudtrail/logs \
```

```
    --filter-name KMS-Key-Disabled \
    --filter-pattern '{($.eventSource=kms.amazonaws.com) && (($.eventName=DisableKey) || ($.ev
    --metric-transformations \
        metricName=KMSKeyDisabled,metricNamespace=CloudTrailMetrics,metricValue=1

aws cloudwatch put-metric-alarm \
    --alarm-name KMS-Key-Disabled-Alert \
    --alarm-description "CRITICAL: KMS key was disabled or scheduled for deletion" \
    --metric-name KMSKeyDisabled \
    --namespace CloudTrailMetrics \
    --statistic Sum \
    --period 60 \
    --evaluation-periods 1 \
    --threshold 1 \
    --comparison-operator GreaterThanOrEqualToThreshold \
    --alarm-actions arn:aws:sns:us-east-1:123456789012:CriticalSecurityAlerts
```

3. Alarmes API et Accès Réseau

3.1 Appels API Non Autorisés

```
# Détecter les appels API rejetés (UnauthorizedOperation, AccessDenied)
aws logs put-metric-filter \
    --log-group-name /aws/cloudtrail/logs \
    --filter-name Unauthorized-API-Calls \
    --filter-pattern '{($.errorCode=*UnauthorizedOperation) || ($.errorCode=AccessDenied*)}' \
    --metric-transformations \
        metricName=UnauthorizedAPICalls,metricNamespace=CloudTrailMetrics,metricValue=1

aws cloudwatch put-metric-alarm \
    --alarm-name Unauthorized-API-Calls-Spike \
    --alarm-description "Spike in unauthorized API calls detected" \
    --metric-name UnauthorizedAPICalls \
    --namespace CloudTrailMetrics \
    --statistic Sum \
    --period 300 \
    --evaluation-periods 1 \
    --threshold 10 \
    --comparison-operator GreaterThanThreshold \
    --alarm-actions arn:aws:sns:us-east-1:123456789012:SecurityAlerts
```

CloudWatch Logs Insights

1. Requêtes de Sécurité Essentielles

1.1 Top 10 Utilisateurs IAM avec le Plus d'Erreurs

```
fields @timestamp, userIdentity.arn, eventName, errorCode, errorMessage
| filter errorCode like /(?i)(denied|unauthorized|forbidden)/
| stats count(*) as errorCount by userIdentity.arn
| sort errorCount desc
| limit 10
```

1.2 Identifier les Accès depuis des Pays Inhabituels

```
fields @timestamp, userIdentity.arn, sourceIPAddress, awsRegion, eventName
| filter sourceIPAddress not like /^10\.|^172\.(1[6-9]|2[0-9]|3[0-1])\.|^192\.168\./
| stats count(*) as accessCount by sourceIPAddress, userIdentity.arn
| sort accessCount desc
| limit 20
```

1.3 Détecter les Créations de Ressources Inhabituelles

```
fields @timestamp, userIdentity.arn, eventName, requestParameters
| filter eventName like /^(Create|Run|Launch)/
| filter eventTime >= ago(24h)
| stats count(*) as resourceCreations by userIdentity.arn, eventName
| sort resourceCreations desc
```

1.4 Exfiltration de Données Potentielle (S3)

```
fields @timestamp, userIdentity.arn, eventName, requestParameters.bucketName, additionalEventD
| filter eventName = "GetObject" and additionalEventData.bytesTransferredOut > 1073741824
| stats sum(additionalEventData.bytesTransferredOut) as totalBytes by userIdentity.arn, reques
| sort totalBytes desc
```

1.5 Activité Hors Heures Ouvrables

```
fields @timestamp, userIdentity.arn, eventName, sourceIPAddress
| filter eventTime >= ago(7d)
| filter strftime("%H", @timestamp) < "08" or strftime("%H", @timestamp) > "18"
| stats count(*) as afterHoursActivity by userIdentity.arn
| sort afterHoursActivity desc
| limit 20
```

2. Analyse VPC Flow Logs

2.1 Top 10 IPs Sources avec Connexions Rejetées

```
fields @timestamp, srcAddr, dstAddr, dstPort, action
| filter action = "REJECT"
| stats count(*) as rejectedConnections by srcAddr, dstPort
| sort rejectedConnections desc
| limit 10
```

2.2 Détecter un Scan de Ports

```
fields @timestamp, srcAddr, dstPort
| filter action = "REJECT"
| stats count_distinct(dstPort) as uniquePorts by srcAddr
| filter uniquePorts > 50
| sort uniquePorts desc
```

2.3 Identifier Data Exfiltration (> 10GB sortant)

```
fields @timestamp, srcAddr, dstAddr, bytes, protocol
| filter dstAddr not like /^10\.|^172\.(1[6-9]|2[0-9]|3[0-1])\.|^192\.168\./
| stats sum(bytes) as totalBytes by srcAddr, dstAddr
| filter totalBytes > 10737418240
| sort totalBytes desc
```

2.4 Connexions vers des Ports Suspects

```
fields @timestamp, srcAddr, dstAddr, dstPort, action
| filter (dstPort = 4444 or dstPort = 1337 or dstPort = 31337 or dstPort = 8888)
| stats count(*) as suspiciousConnections by srcAddr, dstPort
| sort suspiciousConnections desc
```

Détection d'Anomalies

1. CloudWatch Anomaly Detection

1.1 Activer la Détection d'Anomalies sur Métriques

```
# Créer une alarme avec détection d'anomalies
aws cloudwatch put-metric-alarm \
    --alarm-name API-Gateway-Anomaly-Detection \
    --comparison-operator LessThanLowerOrGreaterThanUpperThreshold \
```

```
    --evaluation-periods 2 \
    --metrics '[
      {
        "Id": "m1",
        "ReturnData": true,
        "MetricStat": {
          "Metric": {
            "Namespace": "AWS/ApiGateway",
            "MetricName": "Count",
            "Dimensions": [{"Name": "ApiName", "Value": "MyAPI"}]
          },
          "Period": 300,
          "Stat": "Sum"
        }
      },
      {
        "Id": "ad1",
        "Expression": "ANOMALY_DETECTION_BAND(m1, 2)",
        "Label": "Count (expected)"
      }
    ]' \
    --threshold-metric-id ad1 \
    --alarm-actions arn:aws:sns:us-east-1:123456789012:AnomalyAlerts
```

1.2 Anomalies sur Secrets Manager

```
# Détecter les accès anormaux à Secrets Manager
aws cloudwatch put-metric-alarm \
    --alarm-name Secrets-Manager-Anomaly \
    --comparison-operator LessThanLowerOrGreaterThanUpperThreshold \
    --evaluation-periods 2 \
    --metrics '[
      {
        "Id": "m1",
        "MetricStat": {
          "Metric": {
            "Namespace": "AWS/SecretsManager",
            "MetricName": "ResourceCount"
          },
          "Period": 300,
          "Stat": "Sum"
        }
      },
      {
        "Id": "ad1",
        "Expression": "ANOMALY_DETECTION_BAND(m1, 2)"
      }
    ]' \
    --threshold-metric-id ad1
```

2. CloudWatch Logs Anomaly Detection

2.1 Activer la Détection d'Anomalies sur Logs

```
# Créer un détecteur d'anomalies de logs
aws logs create-log-anomaly-detector \
    --log-group-name /aws/lambda/my-function \
    --anomaly-detector-name lambda-anomaly-detector \
    --evaluation-frequency FIFTEEN_MIN \
    --filter-pattern ""
```

2.2 Types d'Anomalies Détectées

CloudWatch Logs Anomaly Detection utilise le machine learning pour détecter **5 types d'anomalies**:

Type	Description	Exemple
Pattern Frequency	Changement dans la fréquence d'un pattern	Erreur qui apparaît 10x plus souvent
New Pattern	Nouveau pattern jamais vu	Nouveau type d'erreur
Token Variation	Variation dans les tokens	IPs sources inhabituelles
Numerical Variation	Variation dans les valeurs numériques	Latence inhabituelle
Token Sequence	Séquence de tokens inhabituelle	Ordre d'événements anormal

---

Réponse Automatisée avec EventBridge

1. Architecture de Réponse Automatique

```
# Règle EventBridge pour répondre aux alarmes CloudWatch
aws events put-rule \
    --name SecurityAlarmResponse \
    --event-pattern '{
      "source": ["aws.cloudwatch"],
      "detail-type": ["CloudWatch Alarm State Change"],
      "detail": {
        "alarmName": [{
          "prefix": "Security-"
```

```
        }],
        "state": {
          "value": ["ALARM"]
        }
      }
    }'

# Ajouter une Lambda comme target
aws events put-targets \
    --rule SecurityAlarmResponse \
    --targets "Id"="1","Arn"="arn:aws:lambda:us-east-1:123456789012:function:SecurityResponseF
```

2. Lambda de Réponse Automatique

```
import boto3
import json

ec2 = boto3.client('ec2')
ssm = boto3.client('ssm')
sns = boto3.client('sns')

def lambda_handler(event, context):
    """Réponse automatique aux alarmes de sécurité"""

    alarm_name = event['detail']['alarmName']
    alarm_state = event['detail']['state']['value']

    # Root Account Usage Detected
    if alarm_name == "Root-Account-Usage-Detected" and alarm_state == "ALARM":
        response = {
            'action': 'CRITICAL_ALERT',
            'message': 'Root account usage detected - Manual investigation required'
        }

        # Envoyer une alerte critique
        sns.publish(
            TopicArn='arn:aws:sns:us-east-1:123456789012:CriticalSecurityAlerts',
            Subject='CRITICAL: Root Account Usage',
            Message=json.dumps(response, indent=2)
        )

    # Unauthorized API Calls Spike
    elif alarm_name == "Unauthorized-API-Calls-Spike" and alarm_state == "ALARM":
        # Identifier l'IP source depuis CloudTrail
        cloudtrail = boto3.client('cloudtrail')
        events = cloudtrail.lookup_events(
            LookupAttributes=[{
                'AttributeKey': 'EventName',
                'AttributeValue': 'AccessDenied'
            }],
            MaxResults=50
        )
```

```
        # Extraire les IPs suspectes
        suspicious_ips = set()
        for event in events['Events']:
            source_ip = json.loads(event['CloudTrailEvent'])['sourceIPAddress']
            suspicious_ips.add(source_ip)

        # Bloquer les IPs dans le NACL
        for ip in suspicious_ips:
            block_ip_in_nacl(ip)

        response = {
            'action': 'IPS_BLOCKED',
            'blocked_ips': list(suspicious_ips)
        }

        sns.publish(
            TopicArn='arn:aws:sns:us-east-1:123456789012:SecurityAlerts',
            Subject='Automatic Response: IPs Blocked',
            Message=json.dumps(response, indent=2)
        )

    # Security Group Change Detected
    elif alarm_name == "Security-Group-Change-Detected":
        # Exécuter un Systems Manager Automation pour auditer
        ssm.start_automation_execution(
            DocumentName='AWS-AuditSecurityGroupChanges',
            Parameters={
                'AutomationAssumeRole': ['arn:aws:iam::123456789012:role/SecurityAutomation']
            }
        )

    return {
        'statusCode': 200,
        'body': json.dumps('Security response executed')
    }

def block_ip_in_nacl(ip_address):
    """Bloquer une IP dans le NACL"""
    # Trouver un numéro de règle disponible
    nacl_id = 'acl-xxxxx'  # NACL de production

    # Ajouter une règle de blocage
    try:
        ec2.create_network_acl_entry(
            NetworkAclId=nacl_id,
            RuleNumber=get_next_rule_number(nacl_id),
            Protocol='-1',
            RuleAction='deny',
            Egress=False,
            CidrBlock=f'{ip_address}/32'
        )
        print(f"Blocked IP: {ip_address}")
    except Exception as e:
        print(f"Error blocking IP {ip_address}: {e}")
```

```
def get_next_rule_number(nacl_id):
    """Trouver le prochain numéro de règle disponible"""
    response = ec2.describe_network_acls(NetworkAclIds=[nacl_id])
    existing_rules = [entry['RuleNumber'] for entry in response['NetworkAcls'][0]['Entries']]
    # Chercher un numéro entre 100-200 (réservé pour blocages automatiques)
    for rule_num in range(100, 200):
        if rule_num not in existing_rules:
            return rule_num
    return None
```

3. Systems Manager Automation pour Remédiation

```
# Document SSM pour isoler une instance compromise
schemaVersion: '0.3'
description: Isolate compromised EC2 instance
parameters:
  InstanceId:
    type: String
    description: Instance ID to isolate
mainSteps:
  - name: CreateSnapshot
    action: 'aws:executeAwsApi'
    inputs:
      Service: ec2
      Api: CreateSnapshot
      VolumeId: '{{ InstanceId }}'
      Description: 'Forensic snapshot before isolation'
    outputs:
      - Name: SnapshotId
        Selector: $.SnapshotId

  - name: AttachQuarantineSecurityGroup
    action: 'aws:executeAwsApi'
    inputs:
      Service: ec2
      Api: ModifyInstanceAttribute
      InstanceId: '{{ InstanceId }}'
      Groups:
        - sg-quarantine-xxxxx  # Security group qui bloque tout

  - name: SendNotification
    action: 'aws:executeAwsApi'
    inputs:
      Service: sns
      Api: Publish
      TopicArn: 'arn:aws:sns:us-east-1:123456789012:SecurityAlerts'
      Subject: 'Instance Isolated'
      Message: 'Instance {{ InstanceId }} has been isolated. Snapshot: {{ CreateSnapshot.Snaps
```

Contributor Insights pour Analyse Sécurité

1. Top-N IPs Sources avec Connexions Rejetées (VPC Flow Logs)

```
{
  "Schema": {
    "Name": "CloudWatchLogRule",
    "Version": 1
  },
  "AggregateOn": "Count",
  "Contribution": {
    "Filters": [
      {
        "Match": "$.action",
        "EqualTo": "REJECT"
      }
    ],
    "Keys": [
      "$.srcAddr",
      "$.dstPort"
    ]
  },
  "LogFormat": "CLF",
  "LogGroupNames": [
    "/aws/vpc/flowlogs"
  ]
}
```

```
# Créer la règle Contributor Insights
aws cloudwatch put-insight-rule \
    --rule-name VPC-Rejected-Connections \
    --rule-state ENABLED \
    --rule-definition file://contributor-insights-rule.json
```

2. Top-N Utilisateurs avec Appels API Échoués (CloudTrail)

```
{
  "Schema": {
    "Name": "CloudWatchLogRule",
    "Version": 1
  },
  "AggregateOn": "Count",
  "Contribution": {
    "Filters": [
      {
```

```
        "Match": "$.errorCode",
        "In": [
          "AccessDenied",
          "UnauthorizedOperation",
          "InvalidPermission.NotFound"
        ]
      }
    ],
    "Keys": [
      "$.userIdentity.arn",
      "$.eventName"
    ]
  },
  "LogFormat": "JSON",
  "LogGroupNames": [
    "/aws/cloudtrail/logs"
  ]
}
```

Checklist de Supervision

✅ Alarmes Critiques (Priorité 1)

- [ ] **Utilisation du compte root**
- [ ] **Changements de politiques IAM**
- [ ] **Changements de security groups**
- [ ] **Clés KMS désactivées/supprimées**
- [ ] **Échecs de connexion console (> 5 en 5 min)**
- [ ] **Appels API non autorisés (spike)**
- [ ] **Changements de configuration réseau (NACLs)**
- [ ] **Désactivation de CloudTrail**

✅ Logs et Rétention (Priorité 1)

- [ ] **CloudTrail logs dans S3 + CloudWatch Logs**
- [ ] **VPC Flow Logs activés (ALL traffic)**
- [ ] **Logs WAF activés et centralisés**
- [ ] **Rétention > 90 jours (compliance)**
- [ ] **Chiffrement KMS pour tous les log groups**
- [ ] **Intégrité des logs CloudTrail vérifiée**

✅ Détection et Analyse (Priorité 2)

- [ ] **CloudWatch Logs Insights queries documentées**
- [ ] **Anomaly Detection activée (Lambda, API Gateway)**
- [ ] **Contributor Insights rules configurées**
- [ ] **GuardDuty activé et intégré**
- [ ] **Security Hub activé avec standards AWS**
- [ ] **Amazon Detective activé pour investigations**

✅ Réponse Automatisée (Priorité 2)

- [ ] **EventBridge rules pour alarmes critiques**
- [ ] **Lambda de réponse automatique déployées**
- [ ] **Systems Manager Automation runbooks**
- [ ] **SNS topics pour alertes (Email + Slack/PagerDuty)**
- [ ] **Incident Manager configuré**

✅ Audit et Conformité (Priorité 3)

- [ ] **Dashboards CloudWatch pour vue d'ensemble**
- [ ] **Rapports hebdomadaires automatisés**
- [ ] **Métriques de sécurité suivies (KPIs)**
- [ ] **Tests réguliers des alarmes**
- [ ] **Documentation des runbooks**
- [ ] **Formation équipe sur les playbooks**

Références et Ressources

Documentation Officielle AWS

- CloudWatch Best Practices
- CloudWatch Logs Insights Query Syntax
- CloudWatch Anomaly Detection
- Contributor Insights

Outils et Services

- **Amazon CloudWatch** - Monitoring et logs
- **Amazon EventBridge** - Événements et orchestration
- **Amazon GuardDuty** - Détection de menaces
- **AWS Security Hub** - Posture de sécurité
- **Amazon Detective** - Investigation
- **AWS Systems Manager** - Automatisation

Conclusion

Une supervision de sécurité efficace repose sur:

1. **Alarmes proactives** sur les événements critiques (IAM, réseau, accès)
2. **Analyse continue** avec Logs Insights et Contributor Insights
3. **Détection d'anomalies** automatisée par machine learning
4. **Réponse automatisée** via EventBridge et Lambda
5. **Visibilité complète** avec logs centralisés et chiffrés

L'implémentation de ces pratiques permet de **réduire le temps de détection (MTTD) de 70%** et le temps de réponse (MTTR) de **30%** selon les études AWS 2025.

Document préparé pour: [Nom du Client]
Contact support: [Email de l'équipe SRE/Sécurité]
Dernière mise à jour: Novembre 2025