Incidents in the Cloud: deal with them!

8 December 2023 - 7 min. read

AWS Incident Management ***Business Continuity***

"I don't make things complicated. That's the way they get, all by themselves."

- Martin Riggs, Lethal Weapon.

We design architectures by taking care of all of the best practices and we train our collaborators to let them acquire high skills and document everything. BUT… Sometimes something still goes wrong, and we hear or say the word "incident".

But what is an *incident*?

An incident is an event that disrupts or reduces the quality of an IT service or poses a risk to the security or performance of a system. It can be a server outage, a network failure, a malware infection, or a data breach.

We all work to avoid them and be sure to expose ourselves to the minimum risk; however, IT incidents are inevitable, and they can have a significant impact on business performance, reputation, and customer experience. We all have to deal with incidents sometimes in our work careers, and managing them is not an easy skill to acquire.

This article will explain the **key elements and best practices of incident management**.

We will also introduce you to AWS Systems Manager Incident Manager, a feature of AWS Systems Manager that helps to prepare and respond to application and infrastructure incidents.

Incident Management

Incident management is crucial when offering customers a reliable and secure service. It's a process that helps identify, analyze, and resolve any unplanned event or issue affecting quality, availability, or performance experienced by users.

Incidents can have different levels of severity and impact, depending on the type of service, the number of users affected, the duration of the disruption, and the potential consequences. An incident management process aims to restore normal service operativity as quickly as possible.

By implementing an effective incident management process, we can prevent incidents and reduce (or eliminate) downtime by improving our Mean Time To Resolution (MTRR), leading to a better customer experience.

To briefly describe key elements, the incident management process typically consists of the following stages:

- **Detection**: when an incident is discovered or received.
- **Classification**: a priority is assigned based on its severity and impact.
- **Escalation**: the appropriate people and teams are involved to handle the incident based on its priority and category.
- **Diagnosis**: investigating and analyzing the root cause and possible solutions.
- **Resolution**: the best solution is implemented to resolve the incident and verify that the service is restored to normal.
- **Closure**: the phase of documenting and communicating incident details, outcomes, lessons learned, and closing the incident record.

Another critical point is to identify who is responsible for each stage. This depends on the organization's size and structure and the incident's complexity and nature. However, there are some common roles and responsibilities found in most incident management teams, such as:

- **Incident manager**: oversees and coordinates the entire incident management process, from detection to closure.
- **Incident owner**: accountable for the incident, has the authority to make decisions and approve actions related to the incident.
- **Incident team**: the group of people involved in the diagnosis and resolution of the incident, with the skills and expertise to handle the incident.
- **Incident reporter**: detects or reports the incident and provides the initial information and details.

Last but not least, we must find a solution to ease our lives when dealing with incidents. Incident Manager, with its integration with other AWS services, can help in various phases.

First, let's see how it can map stages from the process.

We can define **contacts** and **contact channels** for **engagement plans**, involve the right stakeholders, and keep everyone in sync; this is always the best strategy when dealing with problems. Having clear communication makes the difference. Needless to say, you can also define **on-call schedules**. **Escalation plans** then define escalations in notifications when required.

We can also define **Automation Runbooks** that leverage AWS Systems Manager Automation to automate common tasks to avoid error-prone manual operations. They are also useful for automating incident responses and providing detailed steps to first responders.

A **response plan** links all the elements above, defining what must be in place when an incident occurs, such as who is required to respond, the established automated response, and the collaboration tool to use.

Incidents can then be created in an automated manner, for example, by leveraging EventBridge rules, SecurityHub findings, and CloudWatch alarms.

When an incident happens, AWS Incident Manager automatically gathers data about the AWS resources affected by the incident and shows this data on the Related Items tab. You can also use a runbook in your response plan to help fix the problem. When an incident happens, Incident Manager can pass the data about the affected AWS resources to the runbook. Then, the runbook can use that data to target those resources and try to resolve the issue.

Let's see a simple use case that you can expand and adapt to your needs.

We will monitor a site-to-site VPN; when a tunnel goes down these actions will be performed:

- An incident will be created
- A notification will be sent on a Slack channel
- The AWS-managed troubleshooting SSM document will be executed, so it will perform the initial analysis on CloudWatch Logs Insight to help the team understand the problem.

This is an example template. It will need to be extended to adapt to your configuration.

```
AWSTemplateFormatVersion: 2010-09-09
Description: A template that creates a CloudWatch alarm for a site-to-site VPN connection and an incident response plan for AWS Systems Manager Incident Manager.

Parameters:
  VpnConnectionId:
    Type: String
    Description: The ID of the VPN connection to monitor.
    AllowedPattern: ^vpn-[0-9a-f]{8,17}$
    ConstraintDescription: Must be a valid VPN connection ID.

Resources:
  VpnAlarm:
    Type: AWS::CloudWatch::Alarm
    Properties:
      AlarmName: !Sub "VPN connection ${VpnConnectionId} status alarm"
      AlarmDescription: An alarm that triggers when the VPN connection status is DOWN.
      Namespace: AWS/VPN
      MetricName: TunnelState
      Dimensions:
        - Name: VpnId
          Value: !Ref VpnConnectionId
      Statistic: Minimum
      Period: 60
      EvaluationPeriods: 1
      Threshold: 1
      ComparisonOperator: LessThanThreshold
      AlarmActions:
        - !Ref IncidentResponsePlan

  IncidentResponsePlan:
    Type: AWS::SSMIncidents::ResponsePlan
    Properties:
      Name: !Sub "VPN connection ${VpnConnectionId} incident response plan"
      DisplayName: !Sub "VPN connection ${VpnConnectionId} incident respo
```

```
nse plan"
      ChatChannel:
        ChatbotSns: !Ref NotificationTopic
      IncidentTemplate:
        Title: !Sub "VPN connection ${VpnConnectionId} is DOWN"
        Impact: 3
        Summary: "The VPN connection to the remote site is not working."
        DedupeString: !Sub "VPN connection ${VpnConnectionId} is DOWN"
      Actions:
        - SsmAutomation:
            RoleArn: !GetAtt AutomationRole.Arn
            DocumentName: AWSSupport-TroubleshootVPN
            DocumentVersion: "1"
            Parameters:
              VpnConnectionId: !Ref VpnConnectionId

  NotificationTopic:
    Type: AWS::SNS::Topic
    Properties:
      DisplayName: "VPN connection status notification"
      TopicName: "vpn-connection-status-notification"

  SlackChannelConfiguration:
    Type: AWS::Chatbot::SlackChannelConfiguration
    Properties:
      ConfigurationName: "vpn-connection-status-slack"
      IamRoleArn: !GetAtt ChatbotRole.Arn
      SlackChannelId: "YourChannelID"
      SlackWorkspaceId: "SlackWorkspace"
      SnsTopicArns:
        - !Ref NotificationTopic

  AutomationRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
          - Effect: Allow
            Principal:
```

```yaml
            Service: ssm.amazonaws.com
          Action: sts:AssumeRole
    Path: "/"
    Policies:
      - PolicyName: "vpn-troubleshoot-policy"
        PolicyDocument:
          Version: "2012-10-17"
          Statement:
            - Effect: Allow
              Action:
                - ec2:DescribeVpnConnections
                - ec2:DescribeVpnGateways
                - ec2:DescribeCustomerGateways
                - ec2:ResetVpnConnection
              Resource: "*"

ChatbotRole:
  Type: AWS::IAM::Role
  Properties:
    AssumeRolePolicyDocument:
      Version: "2012-10-17"
      Statement:
        - Effect: Allow
          Principal:
            Service: chatbot.amazonaws.com
          Action: sts:AssumeRole
    Path: "/"
    Policies:
      - PolicyName: "chatbot-policy"
        PolicyDocument:
          Version: "2012-10-17"
          Statement:
            - Effect: Allow
              Action:
                - cloudwatch:DescribeAlarms
                - cloudwatch:ListMetrics
                - cloudwatch:GetMetricData
                - cloudwatch:GetMetricStatistics
                - cloudwatch:PutMetricData
                - ec2:DescribeInstances
```

```
                    - ec2:DescribeRegions
                    - ec2:DescribeVpnConnections
                    - ec2:DescribeVpnGateways
                    - ec2:DescribeCustomerGateways
                    - ec2:ResetVpnConnection
                    - sns:ListTopics
                    - sns:ListSubscriptionsByTopic
                    - sns:Publish
                  Resource: "*"
```

AWS Documentation gives another nice example to monitor and alert if there is an activity done by someone using the root account (showing also some console configuration).

As you can see, you can start using this service and adapt it to your organization's needs because it integrates quickly with all AWS services.

Closing thoughts

Managing incidents is challenging, and over-engineering can slow down this critical process. Our advice is to start with an agile mindset and implement a simple solution that fits the basic needs, then observe the outputs and improve for the future.

When we speak about processes and methods, there is no "one-size-fits-all solution: for example, we found that adding additional steps in the post-incident phase is beneficial: we hold a retrospective involving the team members and all the stakeholders for the workload.

How do you prepare for incidents? Is there some practice that you find helpful when dealing with problems? Let us know in the comments!

About Proud2beCloud

Proud2beCloud is a blog by beSharp, an Italian APN Premier Consulting Partner expert in designing, implementing, and managing complex Cloud infrastructures and advanced services on AWS. Before being writers, we are Cloud Experts working daily with AWS services since 2007. We are hungry readers, innovative builders, and gem-seekers. On Proud2beCloud, we regularly share our best AWS pro tips, configuration insights, in-depth news, tips&tricks, how-tos, and many other resources. Take part in the discussion!

Damiano Giorgi

Ex on-prem systems engineer, lazy and prone to automating boring tasks. In constant search of technological innovations and new exciting things to experience. And that's why I love Cloud Computing! At this moment, the only "hardware" I regularly dedicate myself to is that my bass; if you can't find me in the office or in the band room try at the pub or at some airport, then!