Disaster Recovery Plan for AWS Environment

EC2 Instances and RDS

1. Introduction

1.1 Purpose

This disaster recovery plan outlines the procedures and responsibilities for recovering the AWS environment, including EC2 instances and RDS databases.

1.2 Scope

This plan covers the recovery of EC2 instances and RDS databases hosted in the AWS environment.

2. Risk Assessment

2.1 Identified Risks

- AWS Service Outages:
  - *Example*: Disruption in AWS services due to regional outages, affecting the availability of EC2 instances and RDS databases.
- Data Breaches:
  - *Example*: Unauthorized access to AWS resources, leading to the exposure or theft of sensitive data stored in EC2 instances or RDS databases.
- Accidental Resource Deletion:
  - *Example*: Unintentional deletion of critical EC2 instances or RDS databases, resulting in data loss and service unavailability.
- Network Connectivity Issues:
  - *Example*: Issues with internet connectivity or misconfigurations impacting the communication between EC2 instances and RDS databases.

3. Recovery Strategies

3.1 Backup and Restoration

Backup Strategy:

- Frequency of Backups:
  - Implement automated daily snapshots for EC2 instances and RDS databases.
  - Adjust backup frequency based on the criticality of data and business requirements.
- Storage Location:
  - Store backups in Amazon S3 for durability and accessibility.
  - Utilize AWS Backup for centralized management and policy-based backups.
- Lifecycle Policies:
  - Implement lifecycle policies to manage the retention of backups, ensuring compliance with data retention policies.

Restoration Procedures:

- Snapshot Restoration:
  - In the event of instance or database failure, initiate restoration from the latest snapshot.
  - Ensure proper coordination to minimize downtime during the restoration process.

- Automated Configuration:
    - Utilize Infrastructure as Code (IaC) tools like AWS CloudFormation for automated deployment of EC2 instances and RDS databases.
    - Store configuration templates in version-controlled repositories.
- Backup Testing:
    - Regularly perform testing of backup restoration procedures to verify their reliability.
    - Use AWS Recovery Testing services to simulate real-world scenarios and identify potential issues.

3.2 Multi-Region Deployment

Redundancy and Availability Measures:

- Cross-Region Resource Deployment:
    - Deploy critical resources, such as EC2 instances and RDS databases, across multiple AWS regions.
    - Utilize AWS services like Route 53 for DNS-based load balancing and routing traffic to healthy regions.
- Global Accelerators:
    - Implement AWS Global Accelerator to route traffic over the AWS global network, providing low-latency and high availability.
    - Configure Anycast IP addressing for rapid failover in case of endpoint unavailability.
- Multi-AZ Deployments for RDS:
    - Choose multi-AZ deployments for RDS instances to automatically replicate data to a standby instance in another Availability Zone within the same region.
    - Consider cross-region replicas for additional redundancy.

Failover and Recovery Strategies:

- Active-Active Architecture:
    - Design applications with an active-active architecture, distributing the workload across regions.
    - Leverage AWS Auto Scaling and Elastic Load Balancing for dynamic scaling and load distribution.

- Automated Health Checks and Alerts:
    - Implement automated health checks for critical resources with AWS CloudWatch.
    - Set up alerts to notify administrators of any anomalies or issues, triggering rapid response and intervention.
- Disaster Recovery Drill:
    - Regularly conduct disaster recovery drills that involve simulated failures in one region and the failover of resources to an alternate region.
    - Use the AWS Well-Architected Framework to guide best practices for multi-region deployments.

Documentation and Monitoring:

- Monitoring Across Regions:

    - Implement centralized monitoring solutions that provide visibility into the health and performance of resources across all regions.
    - Utilize AWS CloudTrail and Config for tracking changes and monitoring resource configurations.
- Documentation of Regional Dependencies:
    - Maintain comprehensive documentation detailing dependencies between services and resources in different regions.
    - Use this documentation to guide recovery procedures and improve understanding of cross-regional interactions.

4. Recovery Procedures

4.1 Emergency Response

Immediate Actions:

- Incident Detection:
    - Leverage AWS CloudWatch Alarms, AWS Config, and CloudTrail to monitor the AWS environment for anomalies, irregularities, or potential disasters.
    - Set up automated alerts for critical metrics, such as high latency, resource exhaustion, or unexpected configuration changes.
- Notification and Alerting:
    - Establish clear notification channels using AWS Simple Notification Service (SNS) to promptly alert the incident response team and key stakeholders.
    - Configure alerting policies for different severity levels to ensure timely awareness.
- Incident Response Plan Activation:
    - Activate the incident response plan specific to AWS, defining roles and responsibilities for the incident response team.
    - Ensure that the incident response team is well-versed in AWS-specific procedures.
- AWS Support Engagement:
    - Immediately engage with AWS Support to report the incident, seek assistance, and leverage AWS's expertise in troubleshooting and resolution.
    - Provide AWS Support with relevant information, including incident details, affected resources, and any available logs.
- Isolate Affected Resources:
    - Identify and isolate affected resources to prevent further impact on the environment.
    - Leverage AWS Identity and Access Management (IAM) policies to restrict access to compromised resources.

Initial Damage Assessment:

- AWS CloudTrail and Config Review:
    - Analyze AWS CloudTrail logs and AWS Config snapshots to understand recent changes and potential causes of the incident.
    - Identify unauthorized access, configuration changes, or unexpected resource deletions.
- Resource Status Check:

    - Use AWS Management Console, AWS CLI, or AWS SDKs to check the status of critical resources, including EC2 instances, RDS databases, and networking configurations.
    - Validate the availability and integrity of essential services.
- Communication of Initial Findings:
    - Communicate initial assessment findings to the incident response team, AWS Support, and key stakeholders.
    - Provide real-time updates on the situation, along with recommended actions and timelines for resolution.
- Log Analysis:
    - Conduct in-depth analysis of logs, including VPC Flow Logs and AWS CloudTrail logs, to identify patterns, potential attack vectors, or anomalies.
    - Collaborate with AWS Support to interpret log data and gain insights into the nature of the incident.
- Forensic Investigations:
    - If the incident involves potential security threats or unauthorized access, initiate forensic investigations using AWS forensic tools and procedures.
    - Preserve evidence for legal and compliance purposes.

4.2 EC2 Instance Recovery

Recovery Steps:

- Incident Identification:
    - When an incident affecting EC2 instances is detected, promptly identify the affected instances using AWS CloudWatch Alarms, AWS Config, or other monitoring tools.
- Instance Status Check:
    - Use the AWS Management Console, AWS CLI, or AWS SDKs to check the status of affected EC2 instances.
    - Determine whether instances are in a running, stopped, or terminated state.
- Snapshot or Image Creation:
    - Create an Amazon Machine Image (AMI) of the affected EC2 instances.
    - This involves capturing a point-in-time snapshot of the root volume and additional volumes attached to the instances.
- AMI Validation:
    - Validate the created AMI to ensure it captures the necessary configurations, installed software, and data.
    - Utilize the new AMI for future instance launches.
- Instance Termination (Optional):
    - If instances are in a problematic state and cannot be recovered, consider terminating them.
    - Use caution and verify that the termination is necessary, as it will result in permanent data loss.
- Launch New Instances:
    - Launch new EC2 instances using the created AMI.
    - Choose the appropriate instance type, region, and network settings for the new instances.

- Instance Validation:
  - Validate the functionality of the new instances, ensuring that they operate as expected.
  - Monitor the instances for any performance issues or abnormalities.

Automation and Orchestration:

- AWS CloudFormation:
  - Leverage AWS CloudFormation templates to automate the provisioning of EC2 instances, networking, and associated resources.
  - Use Infrastructure as Code (IaC) principles for consistent and repeatable deployments.
- AWS Systems Manager Automation:
  - Implement AWS Systems Manager Automation documents to automate EC2 instance recovery procedures.
  - Define runbooks that capture the steps needed for recovery and automate their execution.
- AWS Lambda Functions:
  - Create AWS Lambda functions to automate specific recovery tasks, such as attaching EBS volumes, updating security groups, or modifying instance attributes.

4.3 RDS Database Recovery

Recovery Steps:

- Incident Identification:
  - Identify the RDS database instances affected by the incident using AWS CloudWatch Alarms, AWS Config, or other monitoring tools.
- Database Status Check:
  - Use the AWS Management Console, AWS CLI, or AWS SDKs to check the status of the affected RDS instances.
  - Determine whether instances are in an available, stopped, or other states.
- Database Snapshot Creation:
  - Create a snapshot of the affected RDS database for point-in-time recovery.
  - Ensure that the snapshot captures the necessary data and configurations.
- Snapshot Validation:
  - Validate the created snapshot to ensure it contains a consistent and usable copy of the database.
  - Confirm the snapshot's completion before proceeding with recovery.
- RDS Instance Restoration:
  - Restore the RDS instance using the created snapshot.
  - Specify the snapshot identifier and configure other instance settings as needed.
- Data Consistency Checks:
  - Perform data consistency checks on the restored RDS instance to ensure the integrity of the database.
  - Execute queries, validate relationships, and verify the correctness of critical data.

Automation and Orchestration:

- AWS CloudFormation:

    - Utilize AWS CloudFormation templates to automate the provisioning of RDS instances and associated resources.
    - Include snapshot creation and instance restoration as part of the automated processes.
- AWS Systems Manager Automation:
    - Implement AWS Systems Manager Automation documents to automate RDS database recovery procedures.
    - Define runbooks that capture the steps needed for recovery and automate their execution.
- AWS Lambda Functions:
    - Create AWS Lambda functions to automate specific recovery tasks, such as updating database parameters, modifying security groups, or performing post-recovery checks.

Post-Recovery Activities:

- Monitoring and Validation:
    - Monitor the restored RDS instance for performance, resource utilization, and any potential issues.
    - Validate the functionality of applications connected to the database to ensure a seamless return to normal operations.
- Communication of Recovery Status:
    - Communicate the status of RDS database recovery to relevant stakeholders, including IT teams and business units.
    - Provide information on any temporary measures in place and the expected timeline for full restoration.

5. Testing and Maintenance

5.1 Regular Testing

Testing Schedule:

- Frequency of Testing:
    - Perform a full-scale disaster recovery test for the AWS environment at least annually.
    - Schedule more frequent targeted tests for specific AWS services, such as EC2 instances or RDS databases.
- Testing Window:
    - Choose a testing window that minimizes impact on production systems and takes advantage of AWS maintenance windows if needed.
    - Communicate the testing schedule to relevant teams and AWS support.
- Scenario Variation:
    - Rotate through different disaster scenarios specific to AWS, such as regional outages, accidental data deletion, or AWS service disruptions.
    - Ensure scenarios reflect the unique challenges of cloud-based infrastructure.

Testing Procedures:

- AWS Resource Recovery:
    - Simulate the recovery of EC2 instances, RDS databases, and other AWS resources.
    - Validate the accuracy of restored configurations and connections.
- Auto Scaling and Load Balancing Testing:

    - Test the effectiveness of Auto Scaling Groups and load balancers in responding to changes in demand.
    - Assess the automatic scaling of resources based on predefined policies.
- Cross-Region Failover:
    - Simulate a regional outage and test the failover to resources in another AWS region.
    - Validate the ability to redirect traffic and maintain service availability.
- AWS Backup and Restore:
    - Perform tests of AWS backup and restore procedures for EC2 instances and RDS databases.
    - Confirm the integrity of data restored from backups.
- Documentation Review:
    - Review and update documentation during the testing process, including AWS configurations and recovery procedures.
    - Ensure that documentation aligns with the current AWS environment.
- Post-Test Evaluation:
    - Conduct a post-test evaluation with key stakeholders, including cloud architects and operations teams.
    - Analyze the effectiveness of the recovery procedures and identify areas for improvement.

5.2 Updates and Maintenance

Infrastructure Changes:

- CloudTrail and Config Monitoring:
    - Leverage AWS CloudTrail and AWS Config to monitor changes in the AWS environment.
    - Integrate monitoring results into the disaster recovery plan update process.
- Infrastructure as Code (IaC):
    - Utilize Infrastructure as Code (IaC) tools, such as AWS CloudFormation or Terraform, for provisioning and updating AWS resources.
    - Store IaC templates in version-controlled repositories and update them as needed.
- Procedure Updates:
    - Continuous Integration/Continuous Deployment (CI/CD) Integration:
    - Integrate disaster recovery procedures into CI/CD pipelines to ensure that changes to application code trigger updates to recovery processes.
    - Automate the generation of documentation based on CI/CD processes.
- Regular Testing Insights:
    - Use insights from regular testing exercises, especially in the AWS environment, to identify areas for improvement in recovery procedures.
    - Update the plan based on lessons learned from testing experiences.

Documentation and Communication:

- Infrastructure Documentation:
    - Maintain comprehensive documentation of AWS infrastructure, including architecture diagrams, security configurations, and networking setups.
    - Update documentation promptly when changes occur.

- Automated Notifications:
    - Implement automated notifications or alerts for relevant teams whenever changes are made to AWS resources.
    - Include information on the nature of changes and potential impacts on the disaster recovery plan.

Collaboration with AWS Support:

- Regular Consultations:
    - Engage in regular consultations with AWS Support to discuss changes in best practices, new features, and recommended updates to recovery strategies.
    - Incorporate AWS Support recommendations into the plan.