Disaster Recovery & Migrations

Disaster Recovery Overview

- Any event that has a negative impact on a company's business continuity or finances is considered a disaster.
- Disaster Recovery (DR) is about preparing for and recovering from a disaster.

Types of Disaster Recovery

- On-premise to On-premise: traditional DR, often very expensive.
- On-premise to AWS Cloud: hybrid recovery model.
- AWS Cloud Region A to AWS Cloud Region B: cloud-native disaster recovery.

Key Concepts

- **RPO (Recovery Point Objective)**: The maximum acceptable amount of data loss measured in time.
- **RTO (Recovery Time Objective)**: The target time to recover the system after a disaster.

RPO and RTO

- **RPO (Recovery Point Objective)**: Defines the maximum acceptable amount of data loss measured in time. It answers the question: "Up to what point in time could data be recovered if a disaster occurs?"

- **RTO (Recovery Time Objective)**: Defines the maximum acceptable length of time that a system can be down after a failure or disaster. It answers the question: "How quickly must the system be restored after a disaster?"

- **RPO** is focused on **data loss**.

- **RTO** is focused on **downtime**.

Disaster Recovery Strategies

There are different strategies for disaster recovery, depending on the desired Recovery Time Objective (RTO). Listed from slower to faster RTO:

- **Backup and Restore**:

  - The simplest and most cost-effective strategy.
  - Data is backed up and stored (e.g., on Amazon S3).
  - Restore operations are initiated only after a disaster occurs.
  - Highest RTO among all strategies.

- **Pilot Light**:

  - A minimal version of the environment is always running.
  - Core components (like databases) are kept ready.
  - Additional infrastructure is launched only when needed.

- **Warm Standby**:

  - A scaled-down but fully functional environment is running.
  - Can be scaled up quickly to handle production traffic.
  - Lower RTO compared to pilot light.

- **Hot Site / Multi Site Approach**:
    - A full production environment is running in another region or site.
    - Instant failover possible.
    - Offers the lowest RTO, but is the most expensive solution.

Backup and Restore (High RPO)

This is a cost-effective disaster recovery strategy but with a high Recovery Point Objective (RPO). It focuses on regularly backing up data and restoring it only after a disaster has occurred.

Key Components and Services

- **Corporate Data Center to AWS Cloud**:
    - Data can be backed up from on-premise to AWS using **AWS Storage Gateway** or **AWS Snowball** for large transfers.
- **AWS Storage Services**:
    - **Amazon S3**: Used for storing backup data.
    - **Amazon S3 Glacier**: Suitable for long-term archival.
    - **EBS Snapshots**: Regular snapshots for Amazon EC2 volumes.
    - **Amazon RDS Snapshots**: For backup of databases.
    - **RDS Lifecycle Management**: Automates backup and retention.
    - **Redshift Snapshots**: Used for Redshift data warehouse backups.
    - **Amazon Machine Images (AMIs)**: Used to back up EC2 instances.
- **Restore Process**:
    - After a disaster, infrastructure (e.g., Amazon EC2, Amazon RDS) is recreated using the saved backups (snapshots, AMIs, etc.).

This method is simple and affordable, but the recovery process is slower compared to other strategies.

Disaster Recovery – Pilot Light

- A minimal version of the application (the "pilot light") is always running in the cloud.
- Critical components, such as databases, are kept active and synchronized.
- This strategy is similar to Backup and Restore, but it allows for quicker recovery because key systems are already operational.
- The rest of the infrastructure (e.g., application servers) is launched only when needed.

Architecture Overview

- **Corporate Data Center**:
    - Primary site for operations.
- **AWS Cloud**:
    - **Amazon RDS**: Running continuously to keep critical data available.
    - **Data Replication**: Ensures that data is kept in sync between on-premises and the cloud.
    - **Amazon EC2**: Instances are preconfigured but not running.
    - **Amazon Route 53**: Can redirect traffic to AWS in case of a disaster.

Pilot Light offers a balance between cost and recovery speed.

Warm Standby

- A full version of the system is deployed and running in the cloud, but it is scaled down to a minimum capacity.
- In the event of a disaster, the infrastructure can be quickly scaled up to handle full production traffic.
- This approach offers a faster Recovery Time Objective (RTO) compared to Pilot Light, while still controlling costs.

Architecture Overview

- **Corporate Data Center**:
    - Hosts the master database and core operations.
- **AWS Cloud**:
    - **RDS Slave**: A read replica is running and synchronized with the master DB.
    - **Data Replication**: Keeps the standby environment in sync.
    - **Amazon Route 53**: Handles DNS failover to redirect traffic.
    - **Elastic Load Balancer (ELB)**: Distributes traffic across instances.
    - **EC2 Auto Scaling**: Instances are running at minimum capacity but can scale quickly.
    - **App Server and Reverse Proxy**: Preconfigured and operational, ready to scale.
    - **Failover**: Occurs from the on-premise master DB to the cloud standby environment.

Warm Standby is ideal for balancing cost and availability.

Multi Site / Hot Site Approach

- This strategy offers the **lowest RTO**, often in minutes or seconds, but it is also the most **expensive**.
- The entire production environment is running both **on-premise** and in **AWS Cloud** simultaneously.
- Known as an **active-active** setup.

Architecture Overview

- **Corporate Data Center**:
    - Fully operational with application servers and master database.
- **AWS Cloud**:
    - **RDS Slave**: Continuously synchronized with the on-premise master database.
    - **Data Replication**: Ensures both environments remain in sync.
    - **Amazon Route 53**: Performs DNS-based routing and failover.
    - **Elastic Load Balancer (ELB)**: Distributes incoming traffic in AWS.
    - **EC2 Auto Scaling**: Runs at full production scale.
    - **App Server and Reverse Proxy**: Fully functional and ready to serve traffic.
    - **Failover**: Can occur automatically if one site becomes unavailable.
    - **Active-Active**: Both sites (on-premise and AWS) handle traffic simultaneously.

This approach ensures high availability and rapid disaster recovery at the cost of higher operational expenses.

All AWS Multi-Region

- This disaster recovery strategy leverages multiple AWS regions for high availability and fault tolerance.

- Ensures **active-active** configuration with very low Recovery Time Objective (RTO).
- Designed for **mission-critical** applications requiring near-zero downtime.

Architecture Overview

- **AWS Cloud (Region A)**:
    - **EC2 Auto Scaling**: Running at full production capacity.
    - **Elastic Load Balancer (ELB)**: Distributes traffic.
    - **Aurora Global (Master)**: Primary database for read/write operations.
- **AWS Cloud (Region B)**:
    - **EC2 Auto Scaling**: Also at production scale.
    - **Elastic Load Balancer (ELB)**: Distributes traffic within this region.
    - **Aurora Global (Slave)**: Read replica of the master database in Region A.
- **Route 53**:
    - Provides DNS-level routing and automatic failover.
    - Supports routing policies for latency-based, geolocation, or weighted distribution.
- **Data Replication**:
    - Continuous synchronization of data between regions, handled by Aurora Global Database and other replication services.

This setup provides high resilience, global reach, and seamless failover capabilities.

Disaster Recovery Tips

Backup

- Use services like:
    - **EBS Snapshots**, **RDS automated backups**, and **RDS Snapshots**.
    - Regularly push data to **Amazon S3**, **S3 Infrequent Access (IA)**, or **Glacier**.
    - Apply **Lifecycle Policies** and enable **Cross Region Replication** for durability.
- From on-premises, use **AWS Snowball** or **AWS Storage Gateway** for large-scale data transfer.

High Availability

- **Route 53**: Redirect DNS from one region to another in case of failure.
- Use **Multi-AZ** deployments for services like:
    - **RDS**
    - **ElastiCache**
    - **EFS**
    - **S3**
- Establish a **Site-to-Site VPN** as a backup for **Direct Connect**.

Replication

- **RDS Cross-Region Replication** and **Amazon Aurora Global Databases** for multi-region setups.
- Enable **database replication** from on-premises to AWS RDS.
- Use **AWS Storage Gateway** to replicate data.

Automation

- Use **CloudFormation** or **Elastic Beanstalk** to recreate entire environments.
- Configure **CloudWatch alarms** to recover or reboot EC2 instances automatically.
- Write **AWS Lambda** functions for custom recovery workflows.

Chaos Engineering

- Adopt principles of chaos engineering to improve resilience.
- Example: Netflix's **Simian Army** tool randomly terminates EC2 instances to test fault tolerance.

DMS – Database Migration Service

- AWS Database Migration Service (DMS) allows you to **quickly and securely migrate databases to AWS**.
- It is **resilient** and **self-healing**, ensuring minimal disruption.
- The **source database remains operational** during the migration process.

Supported Migration Types

- **Homogeneous migrations**: e.g., Oracle to Oracle.
- **Heterogeneous migrations**: e.g., Microsoft SQL Server to Amazon Aurora.

Features

- **Continuous Data Replication**:
    - Uses **Change Data Capture (CDC)** to keep source and target databases in sync during migration.
- **EC2 Instance Requirement**:
    - An **EC2 instance** must be provisioned to run the DMS replication tasks.

This service is ideal for seamless database transitions with minimal downtime.

DMS Sources and Targets

Supported Sources

- **On-Premises and EC2 Databases**:
    - Oracle
    - Microsoft SQL Server
    - MySQL
    - MariaDB
    - PostgreSQL
    - MongoDB
    - SAP
    - IBM DB2
- **Cloud-Based Sources**:
    - Azure SQL Database
    - Amazon RDS (all engines including Aurora)
    - Amazon S3
    - Amazon DocumentDB

Supported Targets

- **On-Premises and EC2 Databases**:

  - Oracle
  - Microsoft SQL Server
  - MySQL
  - MariaDB
  - PostgreSQL
  - SAP
- **AWS Services**:
  - Amazon RDS
  - Amazon Redshift
  - Amazon DynamoDB
  - Amazon S3
  - Amazon OpenSearch Service
  - Amazon Kinesis Data Streams
  - Amazon DocumentDB
  - Amazon Neptune
  - Amazon ElastiCache Redis
  - Babelfish for Aurora PostgreSQL

AWS DMS supports a wide range of data sources and destinations, making it a versatile tool for cloud database migrations.

AWS Schema Conversion Tool (SCT)

- AWS SCT is used to **convert the database schema** from one engine to another.
- It facilitates migrations where the source and target databases use **different engines**.

Use Cases

- **OLTP (Online Transaction Processing)**:
  - Convert from SQL Server or Oracle to MySQL, PostgreSQL, or Aurora.
- **OLAP (Online Analytical Processing)**:
  - Convert from Teradata or Oracle to Amazon Redshift.

Notes

- Prefer **compute-intensive EC2 instances** to optimize schema conversion tasks.
- SCT is **not needed** when the database engine remains the same.
  - Example: Migrating from on-premises PostgreSQL to Amazon RDS PostgreSQL does not require SCT, since the engine (PostgreSQL) is unchanged.

Integration with DMS

- SCT is typically used in combination with **AWS DMS** for migrating both schema and data when the source and target engines differ.

DMS – Continuous Replication

This setup demonstrates how to perform continuous replication from an on-premises database to AWS using AWS DMS and AWS SCT.

Architecture Overview

- **Corporate Data Center**:
  - **Oracle DB** serves as the source database.
- **AWS Environment (VPC)**:
  - **Public Subnet**: Hosts the AWS DMS Replication Instance, which handles both initial load and ongoing Change Data Capture (CDC).
  - **Private Subnet**: Contains the target database, **Amazon RDS for MySQL**.

Process Flow

1. **Schema Conversion**:
   - Use **AWS Schema Conversion Tool (SCT)** installed on a separate server to convert the source schema (Oracle) into the target schema (MySQL).
2. **Data Migration**:
   - AWS DMS performs an initial **full load** of the data from the source to the target.
   - Continuous updates are handled via **Change Data Capture (CDC)** to ensure near real-time replication.

This architecture enables seamless migration and synchronization from heterogeneous on-premise databases to AWS.

AWS DMS – Multi-AZ Deployment

When Multi-AZ is enabled in AWS Database Migration Service (DMS), the system automatically provisions and maintains a standby replica in a separate Availability Zone (AZ) within the same AWS Region.

Architecture Overview

- **Availability Zone A**:
  - Primary **DMS Replication Instance**.
- **Availability Zone B**:
  - **Standby Replica** of the DMS Replication Instance.
  - Synchronized with the primary via **synchronous replication**.

Advantages

- **Data Redundancy**: Ensures high availability and fault tolerance.
- **Eliminates I/O Freezes**: Keeps the replication process smooth, even during instance failover.
- **Minimizes Latency Spikes**: Improves consistency and reliability of data migration and replication.

Multi-AZ deployment is ideal for production-grade DMS tasks requiring high availability.

RDS & Aurora MySQL Migrations

RDS MySQL to Aurora MySQL

- **Option 1**:

    - Use **DB Snapshots** from RDS MySQL.
    - Restore the snapshot as a new Aurora MySQL DB.
- **Option 2**:
    - Create an **Aurora Read Replica** from your existing RDS MySQL.
    - When the replication lag reaches zero, **promote** the replica to a standalone Aurora DB cluster.
    - This process may **take time** and **incur costs**.

External MySQL to Aurora MySQL

- **Option 1**: S3-Based Migration
    - Use **Percona XtraBackup** to create a backup file.
    - Upload the backup to **Amazon S3**.
    - Create an **Aurora MySQL DB** using the backup from S3.
    - This method is **faster**.
- **Option 2**: mysqldump
    - Create a new **Aurora MySQL DB**.
    - Use the **mysqldump** utility to export data from MySQL and import it into Aurora.
    - This method is **slower** than the S3 approach.

Additional Note

- Use **AWS DMS** if both the source (MySQL) and target (Aurora) databases are **up and running** to perform **continuous replication** during migration.

RDS & Aurora PostgreSQL Migrations

RDS PostgreSQL to Aurora PostgreSQL

- **Option 1**:
    - Use **DB Snapshots** from RDS PostgreSQL.
    - Restore them as a new Aurora PostgreSQL DB.
- **Option 2**:
    - Create an **Aurora Read Replica** from your RDS PostgreSQL instance.
    - When replication lag reaches **zero**, promote the replica to its own Aurora DB cluster.
    - This process can take time and may incur additional **costs**.

External PostgreSQL to Aurora PostgreSQL

- Create a **backup** of the source PostgreSQL database.
- Upload the backup to **Amazon S3**.
- Use the **aws_s3 extension** in Aurora PostgreSQL to import the backup from S3.

Additional Note

- Use **AWS DMS** if both the source and target databases are **running**, allowing for **continuous data replication** during the migration process.

On-Premise Strategy with AWS

VM Import / Export

- Amazon Linux 2 AMI is available as a **VM (.iso format)**.
- Supported virtualization platforms include:
  - VMWare
  - KVM
  - VirtualBox (Oracle VM)
  - Microsoft Hyper-V
- Use cases:
  - **Migrate existing applications** into Amazon EC2.
  - Create a **Disaster Recovery (DR) repository** for on-premises virtual machines.
  - **Export VMs** from EC2 back to on-premises infrastructure if needed.

AWS Application Discovery Service

- Gathers detailed information about **on-premises servers** to help plan cloud migrations.
- Provides:
  - **Server utilization metrics**
  - **Dependency mappings**
- Works in conjunction with **AWS Migration Hub** to track the migration progress.

AWS Database Migration Service (DMS)

- Supports **replication** in the following directions:
  - On-premises → AWS
  - AWS → AWS
  - AWS → On-premises
- Compatible with various database technologies (e.g., Oracle, MySQL, DynamoDB).

AWS Server Migration Service (SMS)

- Enables **incremental replication** of **live on-premises servers** to AWS.
- Facilitates seamless lift-and-shift migrations.

AWS Backup

- Fully managed service
- Centrally manage and automate backups across AWS services
- No need to create custom scripts and manual processes

Supported services

- Amazon EC2 / Amazon EBS
- Amazon S3
- Amazon RDS (all DB engines) / Amazon Aurora / Amazon DynamoDB
- Amazon DocumentDB / Amazon Neptune
- Amazon EFS / Amazon FSx (Lustre & Windows File Server)
- AWS Storage Gateway (Volume Gateway)

Features

- Supports cross-region backups

- Supports cross-account backups

AWS Backup (Continued)

Features

- Supports Point-In-Time Recovery (PITR) for supported services
- On-Demand and Scheduled backups
- Tag-based backup policies
- Backup policies are called Backup Plans

Backup Plan Configuration

- Backup frequency:
  - Every 12 hours
  - Daily
  - Weekly
  - Monthly
  - Custom using cron expression
- Backup window (time period in which backups are initiated)
- Transition to Cold Storage:
  - Options: Never, Days, Weeks, Months, Years
- Retention Period:
  - Options: Always, Days, Weeks, Months, Years

AWS Backup (Architecture Overview)

Steps to Set Up Backups

1. **Create Backup Plan**
   - Define frequency and retention policy
2. **Assign AWS Resources**
   - Supported resources include:
     - EC2
     - EBS
     - RDS
     - DynamoDB
     - Aurora
     - EFS
     - FSx
     - Storage Gateway
     - Amazon S3
     - DocumentDB
     - Neptune

Backup Destination

- Backups are automatically stored in **Amazon S3**

AWS Backup Vault Lock

- Enforces a **WORM** (Write Once Read Many) state for all backups stored in AWS Backup Vault
- Provides an additional layer of defense against:
    - Accidental or malicious delete operations
    - Updates that shorten or alter retention periods
- Once enabled, **even the root user cannot delete backups**

Backup Vault Lock Policy

- Ensures that backups **cannot be deleted**

AWS Application Discovery Service

- Helps plan migration projects by collecting data from on-premises data centers
- Provides server utilization data and dependency mapping, which are critical for migrations

Discovery Methods

- **Agentless Discovery** (using AWS Agentless Discovery Connector)
    - Collects:
        - VM inventory
        - Configuration data
        - Performance history (CPU, memory, disk usage)
- **Agent-based Discovery** (using AWS Application Discovery Agent)
    - Collects:
        - System configuration and performance
        - Running processes
        - Network connection details between systems

Data Visualization

- Collected data is accessible through **AWS Migration Hub**

AWS Application Migration Service (MGN)

- Successor of CloudEndure Migration, replaces AWS Server Migration Service (SMS)
- Provides a **lift-and-shift** (rehost) migration solution to simplify moving applications to AWS
- Converts physical, virtual, and cloud-based servers to run natively on AWS
- Supports a wide range of platforms, operating systems, and databases
- Offers **minimal downtime** and **reduced costs**

Migration Workflow

1. **Source environment**: Corporate data center or any cloud
2. **AWS Replication Agent** installed on source servers
3. **Continuous replication** of disks, OS, applications, and databases to AWS
4. **Staging area** in AWS uses low-cost EC2 instances and EBS volumes
5. **Cutover** to **target EC2 instances** and production EBS volumes

- Fully managed migration process using **Application Migration Service**

VMware Cloud on AWS

- Some customers use **VMware Cloud** to manage their on-premises data centers

- These customers want to **extend their data center capacity to AWS** while continuing to use VMware Cloud software

What is VMware Cloud on AWS?

- A solution that enables seamless integration between on-premises VMware environments and AWS infrastructure

Use Cases

- **Migrate** VMware vSphere-based workloads to AWS
- **Run production workloads** across private, public, and hybrid VMware vSphere-based environments
- Implement a **disaster recovery strategy**

Architecture Overview

- On-Premises vCenter manages vSphere-based environments
- VMware Cloud on AWS allows extending vSphere workloads to AWS
- Integrated with AWS services such as:
  - Amazon EC2
  - Amazon S3
  - Amazon FSx
  - Amazon RDS
  - Amazon Redshift
  - AWS Direct Connect

Transferring Large Amounts of Data into AWS

Scenario

- Need to transfer **200 TB** of data to the cloud
- Internet connection speed: **100 Mbps**

Transfer Methods

Over the Internet / Site-to-Site VPN

- **Immediate setup**
- Transfer time:
  - 200 TB × 1000 GB × 1000 MB × 8 Mb / 100 Mbps
  - = 16,000,000 seconds ≈ **185 days**

Over Direct Connect (1 Gbps)

- **Long setup time** (can take over a month)
- Transfer time:
  - 200 TB × 1000 GB × 8 Gb / 1 Gbps
  - = 1,600,000 seconds ≈ **18.5 days**

Using AWS Snowball

- **Approx. 1 week** for end-to-end transfer
- Can be used in combination with **AWS Database Migration Service (DMS)**

For Ongoing Replication or Transfers

- Use **Site-to-Site VPN** or **AWS Direct Connect** with:
  - **AWS DMS** (Database Migration Service)

- **AWS DataSync**