PROJECT PHASE 5

TEAM MEMBER:

NAME: YEPURU JAYANTH

TEAM MEMBERS:

1. **YEPURU SANTHOSH**
2. **YADLAPALLI MAHESHWAR KRITHIK**
3. **JAGADEESH KUMAR DG**
4. **THIRUNAVUKARASU P**

PROJECT TITLE: Disaster recovery with IBM cloud virtual servers

Phase 5: Project Documentation and Submission

The objectives of implementing disaster recovery with IBM Cloud Virtual Servers are to ensure the availability, resilience, and continuity of your virtualized IT infrastructure in the event of a disaster or unexpected disruption. When utilizing IBM Cloud Virtual Servers for disaster recovery, the specific objectives typically include:

Objective:

The primary objective of the disaster recovery plan (DRP) is to ensure the organization's ability to recover from disruptions quickly and efficiently. This includes both minor incidents, such as hardware failures, and major events like natural disasters or cyberattacks.

Design Thinking Process:

The design thinking process involves the following key steps:

1. Empathize: Understand the needs and concerns of key stakeholders, including business owners, IT teams, and end-users.

2. Define: Clearly outline the problem and set specific goals. This might include defining recovery time objectives (RTOs) and recovery point objectives (RPOs) for critical systems.

3. Ideate: Brainstorm potential solutions and strategies for disaster recovery.

4. Prototype: Develop a preliminary disaster recovery plan with a focus on its key components.

5.Test: Evaluate the prototype through simulations, tabletop exercises, and discussions with stakeholders.

6. Implement: Based on feedback and testing, finalize the DRP for deployment.

Development Phases:

1. **Assessment and Analysis:**
   Identify critical business processes, systems, and data by working closely with department heads and subject matter experts.
   Assess potential risks and vulnerabilities, including environmental factors, cybersecurity threats, and human errors.

Define RTOs (the maximum tolerable downtime) and RPOs (the maximum data loss allowed) for each critical component.

2.Strategy Development:

- Develop a disaster recovery strategy that encompasses various components, such as:

- Backup Solutions: Choose the appropriate backup systems (e.g., on-premises, cloud-based) and configure them to meet RPOs and RTOs.

- Replication Mechanisms:

Determine whether synchronous or asynchronous data replication is more suitable for different systems.

- Failover Procedures:

Define how critical systems can failover to secondary sites or cloud environments.

- Communication and Notification Plans:

Establish communication channels to inform stakeholders during a disaster.

- Resource Allocation:

Plan for resource allocation during recovery, including computing resources and personnel.

3. Implementation:

- Deploy the chosen technologies and systems. This includes configuring backup servers, setting up replication mechanisms, and ensuring that monitoring tools are in place to track the health of the DRP components.

- Continuously monitor and maintain the DRP to ensure it remains up to date and effective.

4.Testing and Validation:

- Regularly conduct disaster recovery tests, such as:

- Full-Scale Tests:

Simulate a complete disaster scenario and test the execution of the DRP from start to finish.

- Partial Tests:

Focus on specific components, like data recovery, server failover, or communication restoration.

- Tabletop Exercises:

Involve stakeholders in discussions to evaluate their roles and responsibilities during a disaster.

- Document the results and assess any issues that arise. Make necessary adjustments based on lessons learned.

Disaster recovery planning and implementation for IBM Virtual Servers is a complex task that depends on your specific requirements, resources, and infrastructure. Below is a simplified Python code example that demonstrates a basic scenario for a disaster recovery simulation with IBM Virtual Servers. This code assumes that you have two IBM Virtual Servers (primary and secondary) and that you want to simulate a failover from the primary to the secondary server.

Please note that this is a basic example for educational purposes and does not cover all aspects of a real-world disaster recovery solution, such as data replication, monitoring, and automatic failover.

In [1]:
```
import time

# Simulate a primary IBM Virtual Server
class PrimaryServer:
    def __init__(self):
        self.status = "Running"
        self.data = "Primary server data"

    def simulate_failure(self):
        self.status = "Failed"
        self.data = "Data loss on the primary server"

# Simulate a secondary IBM Virtual Server
class SecondaryServer:
    def __init__(self):
        self.status = "Running"
        self.data = "Secondary server data"

    def failover(self, primary):
        self.data = primary.data
        primary.status = "Failed"

# Create primary and secondary servers
primary_server = PrimaryServer()
secondary_server = SecondaryServer()

print("Primary Server Status:", primary_server.status)
print("Secondary Server Status:", secondary_server.status)

# Simulate a disaster by failing the primary server
print("\nSimulating a disaster...")
primary_server.simulate_failure()

# Initiate a failover to the secondary server
print("\nInitiating failover...")
secondary_server.failover(primary_server)

# Check the status and data after failover
print("\nPrimary Server Status:", primary_server.status)
print("Secondary Server Status:", secondary_server.status)

print("\nData on the Secondary Server after failover:", secondary_server.data)
```

```
# Simulate a disaster by failing the primary server
print("\nSimulating a disaster...")
primary_server.simulate_failure()

# Initiate a failover to the secondary server
print("\nInitiating failover...")
secondary_server.failover(primary_server)

# Check the status and data after failover
print("\nPrimary Server Status:", primary_server.status)
print("Secondary Server Status:", secondary_server.status)

print("\nData on the Secondary Server after failover:", secondary_server.data)
```

```
Primary Server Status: Running
Secondary Server Status: Running

Simulating a disaster...

Initiating failover...

Primary Server Status: Failed
Secondary Server Status: Running

Data on the Secondary Server after failover: Data loss on the primary server
```

5. Documentation and Training:

- Document the disaster recovery plan comprehensively, including:

- Step-by-step procedures for disaster recovery.

- Contact information for key personnel.

- Inventory of critical systems, data, and applications.

- Provide training to staff responsible for executing the DRP. Ensure they understand their roles and responsibilities.

Disaster Recovery Strategy:

Backup Configuration:

- Backup solutions can vary, including traditional tape backups, disk-based backups, and cloud-based backups. Configuration should involve:

- Backup frequency: Determine how often backups are taken (e.g., hourly, daily).

- Data retention policies: Decide how long backup data is kept.

- Storage locations: Ensure backups are stored securely, ideally offsite.

Replication Setup:

- Real-time or near-real-time data replication involves:
  - Selecting replication technologies that match your RPO requirements.
  - Establishing synchronization between primary and secondary systems.
  - Regularly monitoring replication to detect issues promptly.

Recovery Testing Procedures:

- Test procedures can include:
  - Role-specific actions: Specify what each team member should do in a disaster scenario.
  - Data restoration: Verify that backups and replication mechanisms work as expected.
  - Communication recovery: Ensure that employees and stakeholders can be reached and informed during a disaster.

Business Continuity Assurance:

- The DRP guarantees business continuity through several means:

- Minimizing Downtime: Meeting defined RTOs ensures that critical systems are restored quickly, minimizing business interruptions.

- Data Integrity: Frequent backups and data replication help maintain data consistency.

- Rapid Response: A well-documented plan with defined roles and responsibilities enables a swift response.

- Scalability: Provisions for scaling resources during recovery ensure that the plan can adapt to changing business needs.

- Regular Updates and Testing: Continuously updating and testing the DRP ensures that it remains effective and can evolve to address new threats and requirements.

<u>Documentation:</u>

Creating a detailed disaster recovery plan for IBM Cloud Virtual Servers is a critical task to ensure the continuity of your business operations. Below, I'll provide a comprehensive outline for documentation. This documentation should be customized to fit your specific needs and requirements.

Disaster Recovery Plan for IBM Cloud Virtual Servers

Table of Contents

Record the names, IP addresses, and locations of Virtual Servers
Backup and Snapshot Configuration

Describe the backup and snapshot mechanisms provided by IBM Cloud for Virtual Servers

Specify the backup frequency and retention policies

Detail which Virtual Servers are included in the backup and snapshot schedule

Replication Strategy

Document the strategy for ensuring data redundancy and high availability, including:

Built-in replication options

Third-party solutions

Specify data synchronization frequency

Failover Procedures

Document communication channels for notifying key personnel during a disaster

Specify contact lists, roles, and escalation procedures for different disaster scenarios

In [20]:
```python
class DisasterRecoveryPlan:
    def __init__(self):
        self.backup_procedures = {}  # Initialize backup procedures as an empty dictionary
        self.recovery_procedures = {}  # Initialize recovery procedures as an empty dictionary

    def add_backup_procedure(self, component, procedure):
        # Add a backup procedure for a specific component
        self.backup_procedures[component] = procedure

    def add_recovery_procedure(self, component, procedure):
        # Add a recovery procedure for a specific component
        self.recovery_procedures[component] = procedure

    def execute_backup_procedure(self, component):
        # Execute the backup procedure for a specific component
        if component in self.backup_procedures:
            return self.backup_procedures[component]()
        else:
            return f"No backup procedure defined for {component}"

    def execute_recovery_procedure(self, component):
        # Execute the recovery procedure for a specific component
        if component in self.recovery_procedures:
            return self.recovery_procedures[component]()
        else:
            return f"No recovery procedure defined for {component}"

# Instantiate the DisasterRecoveryPlan
drp = DisasterRecoveryPlan()

# Define backup and recovery procedures for IBM Virtual Cloud services
def ibm_virtual_cloud_backup():
    # Implement backup procedure for IBM Virtual Cloud
    return "Backing up data to IBM Virtual Cloud..."

def ibm_virtual_cloud_recovery():
    # Implement recovery procedure for IBM Virtual Cloud
    return "Restoring data from IBM Virtual Cloud..."

# Add IBM Virtual Cloud procedures to the disaster recovery plan
drp.add_backup_procedure("IBM Virtual Cloud", ibm_virtual_cloud_backup)
drp.add_recovery_procedure("IBM Virtual Cloud", ibm_virtual_cloud_recovery)
```

Resource Allocation

Outline how resources, including computing capacity, storage, and network bandwidth, are allocated during recovery

Define resource scaling procedures to accommodate increased demand during recovery

<u>Describe the procedures for regularly testing the disaster recovery plan:</u>

Include the frequency and types of tests (e.g., full-scale disaster simulations, partial component tests)

Document test results and lessons learned

Cloud Service Configuration:

Document the configuration of IBM Cloud services, including Virtual Servers, Load Balancers, and storage options

Specify the cloud region(s) and availability zones being used

Security and Compliance

Highlight security measures and compliance considerations related to disaster recovery,

```
        # Execute the recovery procedure for a specific component
        if component in self.recovery_procedures:
            return self.recovery_procedures[component]()
        else:
            return f"No recovery procedure defined for {component}"

# Instantiate the DisasterRecoveryPlan
drp = DisasterRecoveryPlan()

# Define backup and recovery procedures for IBM Virtual Cloud services
def ibm_virtual_cloud_backup():
    # Implement backup procedure for IBM Virtual Cloud
    return "Backing up data to IBM Virtual Cloud..."

def ibm_virtual_cloud_recovery():
    # Implement recovery procedure for IBM Virtual Cloud
    return "Restoring data from IBM Virtual Cloud..."

# Add IBM Virtual Cloud procedures to the disaster recovery plan
drp.add_backup_procedure("IBM Virtual Cloud", ibm_virtual_cloud_backup)
drp.add_recovery_procedure("IBM Virtual Cloud", ibm_virtual_cloud_recovery)

# Execute backup and recovery procedures for IBM Virtual Cloud
print(drp.execute_backup_procedure("IBM Virtual Cloud"))
print(drp.execute_recovery_procedure("IBM Virtual Cloud"))
```

```
Backing up data to IBM Virtual Cloud...
Restoring data from IBM Virtual Cloud...
```

including encryption, access controls, and regulatory requirements
Regular Maintenance and Updates

Explain how the disaster recovery plan is kept up-to-date to accommodate changing business needs and emerging threats
Include a schedule for plan reviews and updates
Contact Information

Provide contact information for key personnel responsible for disaster recovery, both within your organization and at IBM Cloud Support
Appendices

Include supplementary materials, such as network diagrams, configuration files, and additional technical details
Glossary

Include a glossary of technical terms and acronyms for reference
This documentation should be periodically reviewed and updated to ensure that it remains current and effective in addressing your

organization's disaster recovery needs. Collaborate with relevant stakeholders and subject matter experts to make sure the plan aligns with your specific business and technical requirements.