Continuing Disaster Recovery Plan Implementation

Step 4: Configure Replication to IBM Cloud Virtual Servers

Objective: Implement replication of data and virtual machine images from on-premises systems to IBM Cloud Virtual Servers.

1. **Select Replication Method:**
    - Choose an appropriate replication method that aligns with the disaster recovery strategy and technology stack.
    - Determine whether synchronous or asynchronous replication best suits the RPO established earlier.
2. **Replication Setup:**
    - Configure replication tools or mechanisms to synchronize data and virtual machine images from on-premises to IBM Cloud Virtual Servers.
    - Ensure that replication occurs at defined intervals or in real-time, depending on RPO requirements.
3. **Network and Security Considerations:**
    - Address network and security configurations to enable seamless data transfer and secure replication across on-premises and IBM Cloud environments.
    - Implement encryption and access controls to protect replicated data during transmission.
4. **Monitoring and Validation:**
    - Set up monitoring tools to oversee the replication process, ensuring it operates as intended.
    - Regularly validate replicated data to confirm its accuracy and consistency with the source systems.

Step 5: Conduct Recovery Tests

Objective: Validate the disaster recovery plan by simulating a disaster scenario and practicing recovery procedures.

1. **Test Scenario Development:**
    - Create a range of disaster scenarios that could potentially impact business operations, such as server failure, data corruption, or natural disasters.
    - Define the scope and objectives of each scenario to test different aspects of the recovery plan.
2. **Recovery Procedure Practice:**
    - Execute the recovery procedures outlined in the disaster recovery plan to respond to the simulated disaster scenarios.
    - Involve relevant teams and stakeholders in the recovery exercise to ensure a comprehensive test.
3. **Assessment and Analysis:**
    - Monitor and record the time taken to execute recovery procedures, noting any challenges or bottlenecks encountered.
    - Evaluate the success of the recovery tests against predefined objectives, including meeting RTOs and RPOs.
4. **Iterative Improvement:**

- Analyze the test results and identify any shortcomings or areas for improvement in the disaster recovery plan.
- Make necessary adjustments to enhance the plan's effectiveness and resilience based on test findings.

```
from ibm_cloud import IBMCloudClient


# Connect to IBM Cloud Virtual Servers
ibm_cloud_virtual_server = IBMCloudClient(api_key='YOUR_API_KEY', service='virtual_server')


# Function to configure replication to IBM Cloud Virtual Servers
def configure_replication(source_server, target_virtual_server):
    # Here, 'source_server' refers to your on-premises server or another existing server
    # 'target_virtual_server' refers to the IBM Cloud Virtual Server you want to replicate to

    replication_config = {
        'source_server': source_server,
        'target_virtual_server': target_virtual_server,
        # Add other necessary parameters like replication method, frequency, etc.
    }

    # Call the IBM Cloud SDK function to set up the replication
    ibm_cloud_virtual_server.configure_replication(replication_config)


# Execute the function with required data
configure_replication(source_server='on_premises_server',
target_virtual_server='ibm_cloud_virtual_server')
```

```
from ibm_cloud import IBMCloudClient

# Connect to IBM Cloud Virtual Servers
ibm_cloud_virtual_server = IBMCloudClient(api_key='YOUR_API_KEY', service='virtual_server')

# Simulate a disaster scenario and perform recovery testing
def simulate_and_test_recovery(disaster_scenario):
    # Simulate disaster scenarios (e.g., network failure, server crash)
    if disaster_scenario == 'network_failure':
        # Code to simulate a network failure
        # Trigger recovery procedures
        recovery_process.execute()

    # You can add more conditions for various disaster scenarios and recovery testing

# Execute recovery testing for a specific disaster scenario
simulate_and_test_recovery('network_failure')
```