Disaster Recovery with IBM Cloud Virtual Servers

**Phase 3**

# Development Part 1:

## 1. Review Disaster Recovery Plan:

Before starting the development phase, ensure that you have a clear understanding of disaster recovery plan created in Phase 1. Make sure you have all the documentation, objectives, and designs from the previous phases.

## 2. Setting Up the Environment:

IBM Cloud Virtual Servers: Since disaster recovery plan involves safeguarding business operations with IBM Cloud Virtual Servers, you need to set up virtual servers if they're not already in place. This may involve configuring the servers, allocating resources, and ensuring they are ready for use.

## 

## 3. Data Backup Configuration:

Review the backup configuration section from Phase 1. Ensure you have chosen a suitable backup solution that is compatible with IBM Cloud Virtual Servers.

Set up regular backups for onpremises virtual machine. This includes defining what data and configurations need to be included in backups.

Make sure you have chosen IBM Backup or an equivalent solution to protect missioncritical workloads. Configure it properly to back up data assets.

## 4. Replication Setup:

Review the replication setup section from Phase 1. You should have selected an appropriate replication method, such as blocklevel replication.

Implement the chosen replication method, ensuring data and virtual machine images are replicated to IBM Cloud Virtual Servers. Establish a secure connection between onpremises infrastructure and the IBM Cloud.

## 5. Testing and Validation:

Develop a test plan outlining different recovery scenarios and objectives. This will involve defining the specific tests you want to conduct to validate recovery process.

Execute the test scenarios to validate the effectiveness of disaster recovery plan. This is essential to ensure that plan works as intended and can minimize downtime.

Document the results of the tests and identify any issues or shortcomings in plan. You should aim to address these issues to improve the reliability of disaster recovery plan.

## 6. Business Continuity Integration:

Ensure that disaster recovery plan aligns with organization's broader business continuity strategy. Coordinate with relevant stakeholders to seamlessly integrate the disaster recovery plan into the overall strategy.

## 7. Documentation:

Throughout the development process, maintain detailed documentation of all the steps you've taken. This documentation will be crucial for tracking progress and for any future audits or assessments.

## 8. Communication:

Keep open lines of communication with team members and stakeholders throughout the development phase. Regular updates and clear communication will help ensure everyone is on the same page and that the project is progressing smoothly.

Certainly, I can provide a basic outline for coding in Phase 3 of IBM Disaster Recovery project. However, keep in mind that the actual code may vary depending on specific environment, programming languages, and tools you're using. Below is a simplified example of the steps involved in implementing disaster recovery plan using as an example programming language:

# Phase 3: Development Part 1

## 1. Environment Setup:

Ensure you have access to the IBM Cloud Virtual Servers. You may need to install and configure the necessary cloud SDK or API libraries.

## Code:

### # Example code to connect to IBM Cloud

### from ibm\_cloud import IBMCOS

### cos = IBMCOS(api\_key\_id='\_API\_KEY', endpoint\_url='\_ENDPOINT\_URL')

## 2. Data Backup Configuration:

Set up a script to create regular backups of onpremises virtual machine.

### # Example code to create a backup using IBM Backup service

### backup = cos.create\_backup(source='\_vm', destination='backup\_bucket')

## 3. Replication Setup:

Implement the chosen replication method, such as blocklevel replication, to sync data and virtual machine images with IBM Cloud Virtual Servers.

## Code:

### # Example code for blocklevel replication

### def replicate\_block\_data(source\_vm, destination\_cloud):

### # Implement replication logic here

## 4. Testing and Validation:

Create test scenarios and test scripts to validate the recovery process. This may involve simulating disaster scenarios, restoring data, and ensuring minimal downtime.

## Code:

### # Example code for running a recovery test

### def run\_recovery\_test(scenario):

### # Implement test logic here

## 

## 

## 

## 5. Business Continuity Integration:

Integrate the disaster recovery plan with the organization's business continuity strategy.

## Code:

### # Example code for integrating with business continuity strategy

### def integrate\_with\_bc\_strategy(dr\_plan, bc\_strategy):

### # Implement integration logic here

## 6. Documentation:

Maintain detailed documentation as you code. This may include comments in code and separate documentation files.

## Code:

### # Example code to add comments

### # This function replicates blocklevel data to IBM Cloud

7. Communication:

Ensure regular communication with team members and stakeholders. This might involve sending progress reports via email or communication channels.

Code:

### # Example code to send email updates

### def send\_progress\_email(recipient, message):

### # Implement email sending logic here

## Overall Code:

import time

# Function to set up the environment

def setup\_environment():

# Connect to IBM Cloud Virtual Servers

# This might involve setting up credentials, APIs, and SDKs

# Example: code to connect to IBM Cloud

# Function to perform data backup

def perform\_backup():

# Choose a suitable backup solution compatible with IBM Cloud Virtual Servers

# Implement backup configuration

# Ensure critical data and configurations are included in backups

# Example: code to initiate backups

# Function to set up replication

def setup\_replication():

# Implement replication of data and virtual machine images to IBM Cloud Virtual Servers

# Select the appropriate replication method (e.g., blocklevel replication)

# Establish a secure connection between onpremises infrastructure and IBM Cloud

# Example: code to set up replication

# Function to perform recovery testing

def perform\_recovery\_testing():

# Develop a test plan outlining scenarios and objectives

# Execute test scenarios to validate the recovery process

# Document results, identify issues, and make necessary adjustments

# Example: code to run recovery tests

# Function to integrate with business continuity strategy

def integrate\_with\_bc\_strategy():

# Ensure alignment with the organization's business continuity strategy

# Coordinate with relevant stakeholders to integrate disaster recovery plan

# Ensure compliance with any regulatory or compliance requirements

# Example: code to integrate with business continuity

# Main function to execute Phase 3

def main():

setup\_environment()

perform\_backup()

setup\_replication()

perform\_recovery\_testing()

integrate\_with\_bc\_strategy()

if \_\_name\_\_ == "\_\_main\_\_":

main()

## Conclusion:

In summary, IBM Disaster Recovery project involves the development and coding of the disaster recovery plan. It encompasses reviewing the plan, setting up the environment, configuring backups, implementing data replication, testing the plan's effectiveness, integrating it with the broader business continuity strategy, maintaining documentation, and ensuring open communication with the team. The coding steps provided are a simplified example for implementing the plan, with the understanding that actual code may vary depending on specific requirements and tools used.