IBM Cloud Application

Disaster Recovery with IBM Cloud Virtual Servers

Phase 5

Project Objective:

The IBM Disaster Recovery project's primary objective is to establish a robust disaster recovery plan using IBM Cloud Virtual Servers. This plan guarantees business continuity, especially for onpremises virtual machines, through the definition of recovery objectives, efficient backup configurations, robust replication setup, rigorous recovery testing, and minimizing downtime.

Design Thinking Process:

The project employs a structured and innovative approach to crafting a disaster recovery plan:

1. Disaster Recovery Strategy:

Precisely defining Recovery Time Objective (RTO) and Recovery Point Objective (RPO) parameters.

Continuously evaluating and adapting the plan based on realworld disaster recovery experiences.

2. Backup Configuration:

Choosing a suitable backup solution compatible with IBM Cloud Virtual Servers, emphasizing comprehensive data protection and minimal infrastructure requirements.

3. Replication Setup:

Implementing strong replication methods to ensure data consistency, especially during recovery.

Establishing secure and reliable connections between onpremises infrastructure and IBM Cloud for efficient data transfer.

4. Recovery Testing:

A thorough approach to recovery testing, involving the creation of detailed test scenarios and objectives.

Rigorous execution of scenarios to validate the recovery plan's effectiveness.

Comprehensive documentation of test results and prompt issue resolution for enhanced reliability.

5. Business Continuity Integration:

Seamlessly integrating the disaster recovery plan with the organization's broader business continuity strategy.

Collaborating with stakeholders and adhering to regulatory and compliance requirements to enhance overall resilience.

Development Phases:

The project advances through several development phases:

- **Phase 1: Problem Definition and Design Thinking**

Defining key components of the disaster recovery strategy.
Introducing RTO and RPO as vital parameters.
Laying the groundwork for backup configuration, replication setup, recovery testing, and business continuity integration.

- **Phase 2: Recovery Time Objective (RTO) and Recovery Point Objective (RPO)**

Highlighting the precision in defining RTO to meet modern business expectations.
Emphasizing the importance of minimizing data loss through dynamic RPO criteria.

- **Phase 3: Development Part 1 with Coding**

Exploring the selection of a suitable backup solution aligning with IBM Cloud Virtual Servers.
Focusing on comprehensive data protection while streamlining infrastructure requirements.

Overall Code:

```
import time
```

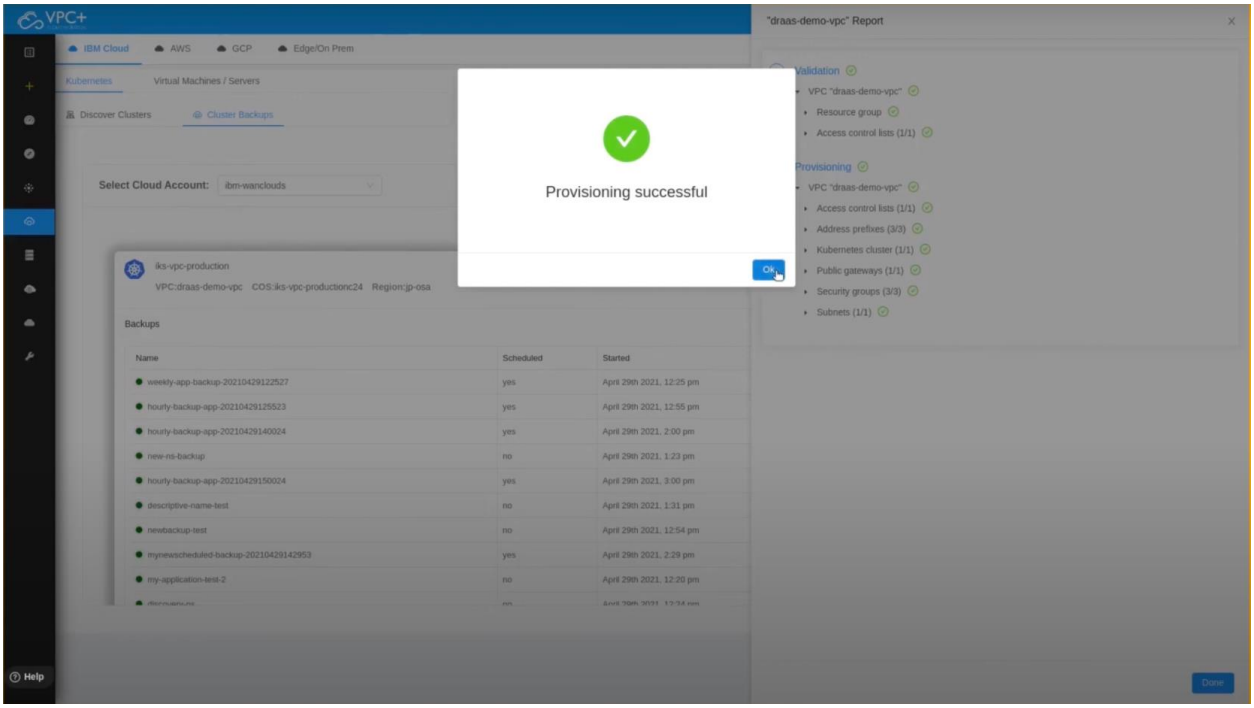
```
# Function to set up the environment
def setup_environment():
    # Connect to IBM Cloud Virtual Servers
    # This might involve setting up credentials, APIs, and SDKs
    # Example:  code to connect to IBM Cloud

# 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
```

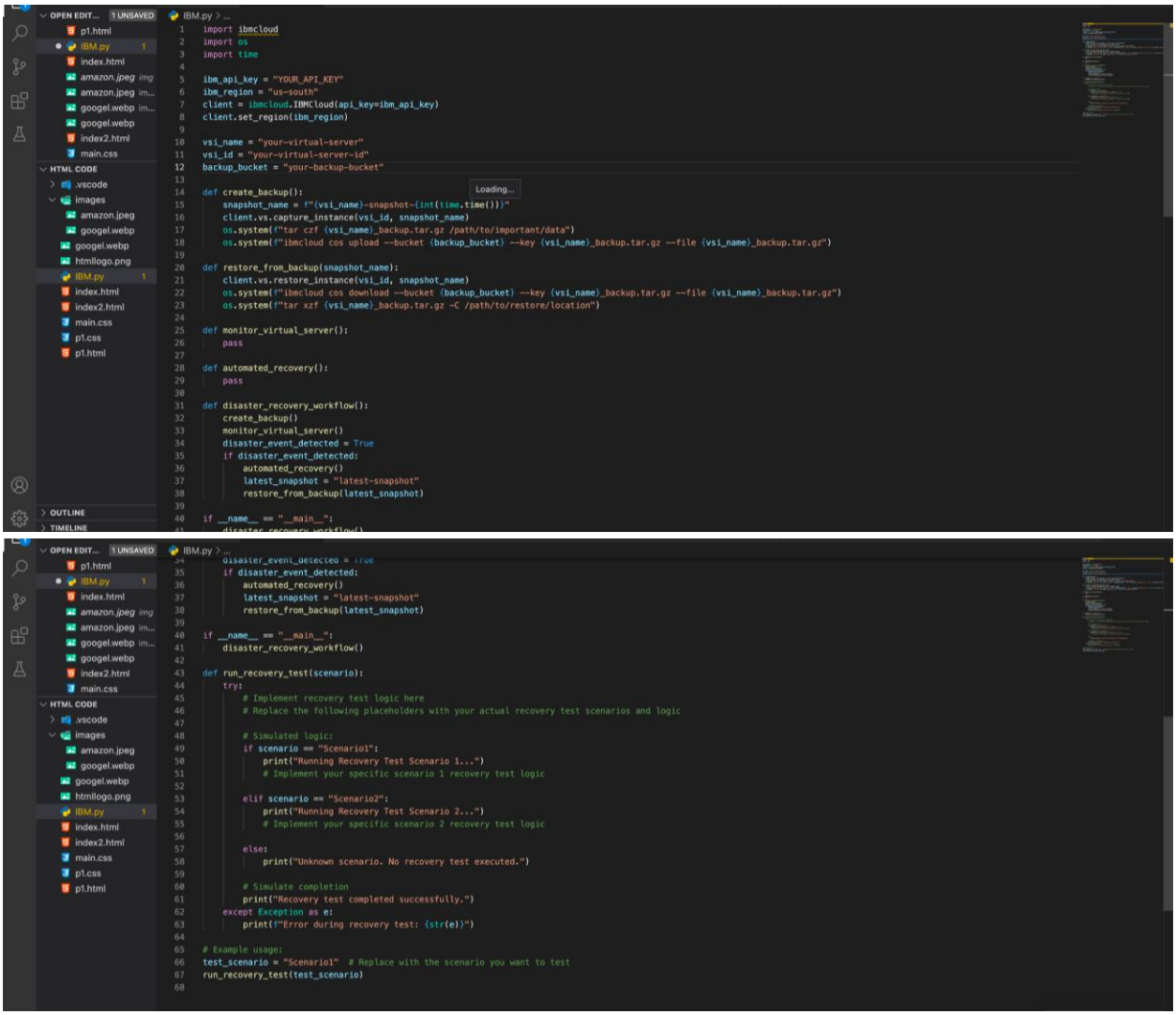
```
# Function to integrate with business continuity strategy
def integrate_with_bc_strategy():
    # Ensure alignment with the organization's business continuity strategy
    # Coordinate with relevant stakeholders to integrate disaster recovery plan
    # Ensure compliance with any regulatory or compliance 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()
```

Phase 4: Development Part 2 with Implementation

Detailing the selection of appropriate replication methods, emphasizing data consistency and reliability.

Discussing the importance of efficient data synchronization between onpremises infrastructure and IBM Cloud.


VPC+
IBM Cloud
AWS
GCP
Edge/On Prem
Kubernetes
Virtual Machines / Servers
Discover Clusters
Cluster Backups
Select Cloud Account:
ibm-wanclouds
Provisioning successful
Ok
iks-vpc-production
VPC:draas-demo-vpc COS:iks-vpc-productionc24 Region:jp-osa
Backups
Name
Scheduled
Started
weekly-app-backup-20210429122527
yes
April 29th 2021, 12:25 pm
hourly-backup-app-20210429125523
yes
April 29th 2021, 12:55 pm
hourly-backup-app-20210429140024
yes
April 29th 2021, 2:00 pm
new-ns-backup
no
April 29th 2021, 1:23 pm
hourly-backup-app-20210429150024
yes
April 29th 2021, 3:00 pm
descriptive-name-test
no
April 29th 2021, 1:31 pm
newbackup-test
no
April 29th 2021, 12:54 pm
mynewscheduled-backup-20210429142953
yes
April 29th 2021, 2:29 pm
my-application-test-2
no
April 29th 2021, 12:20 pm
Help
"draas-demo-vpc" Report
Validation
VPC "draas-demo-vpc"
Resource group
Access control lists (1/1)
Provisioning
VPC "draas-demo-vpc"
Access control lists (1/1)
Address prefixes (3/3)
Kubernetes cluster (1/1)
Public gateways (1/1)
Security groups (3/3)
Subnets (1/1)
Done



```python
import ibmcloud
import os
import time

ibm_api_key = "YOUR_API_KEY"
ibm_region = "us-south"
client = ibmcloud.IBMCloud(api_key=ibm_api_key)
client.set_region(ibm_region)

vsi_name = "your-virtual-server"
vsi_id = "your-virtual-server-id"
backup_bucket = "your-backup-bucket"

def create_backup():
    snapshot_name = f"{vsi_name}-snapshot-{int(time.time())}"
    client.vs.capture_instance(vsi_id, snapshot_name)
    os.system(f"tar czf {vsi_name}_backup.tar.gz /path/to/important/data")
    os.system(f"ibmcloud cos upload --bucket {backup_bucket} --key {vsi_name}_backup.tar.gz --file {vsi_name}_backup.tar.gz")

def restore_from_backup(snapshot_name):
    client.vs.restore_instance(vsi_id, snapshot_name)
    os.system(f"ibmcloud cos download --bucket {backup_bucket} --key {vsi_name}_backup.tar.gz --file {vsi_name}_backup.tar.gz")
    os.system(f"tar xzf {vsi_name}_backup.tar.gz -C /path/to/restore/location")

def monitor_virtual_server():
    pass

def automated_recovery():
    pass

def disaster_recovery_workflow():
    create_backup()
    monitor_virtual_server()
    disaster_event_detected = True
    if disaster_event_detected:
        automated_recovery()
        latest_snapshot = "latest-snapshot"
        restore_from_backup(latest_snapshot)

if __name__ == "__main__":
    disaster_recovery_workflow()

def run_recovery_test(scenario):
    try:
        # Implement recovery test logic here
        # Replace the following placeholders with your actual recovery test scenarios and logic

        # Simulated logic:
        if scenario == "Scenario1":
            print("Running Recovery Test Scenario 1...")
            # Implement your specific scenario 1 recovery test logic

        elif scenario == "Scenario2":
            print("Running Recovery Test Scenario 2...")
            # Implement your specific scenario 2 recovery test logic

        else:
            print("Unknown scenario. No recovery test executed.")

        # Simulate completion
        print("Recovery test completed successfully.")
    except Exception as e:
        print(f"Error during recovery test: {str(e)}")

# Example usage:
test_scenario = "Scenario1"  # Replace with the scenario you want to test
run_recovery_test(test_scenario)
```

Phase 5: Documenting all the Resources and Workflows

Underlining the meticulous design of test scenarios and clear objectives.

Stressing the significance of rigorous execution of recovery tests, documentation, issue resolution, and continuous improvement.

Overall Code:

```
import ibmcloud
import os
import time

ibm_api_key = "YOUR_API_KEY"
ibm_region = "us-south"
client = ibmcloud.IBMCloud(api_key=ibm_api_key)
client.set_region(ibm_region)

vsi_name = "your-virtual-server"
vsi_id = "your-virtual-server-id"
backup_bucket = "your-backup-bucket"

def create_backup():
    snapshot_name = f"{vsi_name}-snapshot-{int(time.time())}"
    client.vs.capture_instance(vsi_id, snapshot_name)
    os.system(f"tar czf {vsi_name}_backup.tar.gz
/path/to/important/data")
    os.system(f"ibmcloud cos upload --bucket {backup_bucket} --key
{vsi_name}_backup.tar.gz --file {vsi_name}_backup.tar.gz")

def restore_from_backup(snapshot_name):
    client.vs.restore_instance(vsi_id, snapshot_name)
```

```
    os.system(f"ibmcloud cos download --bucket {backup_bucket} --
key {vsi_name}_backup.tar.gz --file {vsi_name}_backup.tar.gz")
    os.system(f"tar xzf {vsi_name}_backup.tar.gz -C
/path/to/restore/location")

def monitor_virtual_server():
    pass

def automated_recovery():
    pass

def disaster_recovery_workflow():
    create_backup()
    monitor_virtual_server()
    disaster_event_detected = True
    if disaster_event_detected:
        automated_recovery()
        latest_snapshot = "latest-snapshot"
        restore_from_backup(latest_snapshot)

if __name__ == "__main__":
    disaster_recovery_workflow()

def run_recovery_test(scenario):
    try:
        # Implement recovery test logic here
        # Replace the following placeholders with your actual recovery
test scenarios and logic
```

```
        # Simulated logic:
        if scenario == "Scenario1":
            print("Running Recovery Test Scenario 1...")
            # Implement your specific scenario 1 recovery test logic

        elif scenario == "Scenario2":
            print("Running Recovery Test Scenario 2...")
            # Implement your specific scenario 2 recovery test logic

        else:
            print("Unknown scenario. No recovery test executed.")

        # Simulate completion
        print("Recovery test completed successfully.")
    except Exception as e:
        print(f"Error during recovery test: {str(e)}")

# Example usage:
test_scenario = "Scenario1"  # Replace with the scenario you want to test
run_recovery_test(test_scenario)
```

Conclusion:

In conclusion, this project sets a high standard for disaster recovery planning. By ensuring the alignment of RTO and RPO with modern business expectations, it guarantees data protection and operational continuity while reducing infrastructure requirements. The well-structured development phases and attention to testing and documentation contribute to the plan's reliability. Seamless integration with the broader business continuity strategy further enhances the organization's resilience, ensuring that it can effectively navigate unexpected disasters and disruptions.