**CLOUD APPLICATION DEVELOPMENT**

**PROJECT TITLE : DISASTER RECOVERY WITH IBM CLOUD VIRTUAL SERVERS**

**PHASE 5 : PROJECT DOCUMENTATION AND SUBMISSION**

In this final phase , we are going to document the entire process of our Disaster recovery project , which is DISASTER RECOVERY WITH IBM CLOUD VIRTUAL SERVERS .

So the project consists of four phases : **design thinking , innovation , development , testing and replication.**

**Phase 1: Design Thinking**

The disaster recovery project aims to ensure the uninterrupted availability of critical systems and data in the face of unforeseen disruptions. This documentation outlines the four essential phases of our disaster recovery strategy: Design Thinking, Innovation, Development, and Testing and Replication.

Our organisation faces a range of potential disasters, including natural disasters, cyberattacks, and system failures, which could result in data loss, downtime, and financial losses. Therefore, implementing a robust disaster recovery solution is imperative.

The task goals to increase a catastrophe recovery plan for protecting data and making sure enterprise continuity in emergencies. It involves assessing risks, enforcing information backup and replication, creating a failover system, and establishing a comprehensive catastrophe recuperation plan.

The task has a fundamental intention of ensuring the continuing functionality and resilience of a corporation's important IT infrastructure, especially focusing on IBM Cloud Virtual Servers, within the face of sudden and probably disruptive activities, often known as disasters. These disasters may want to encompass a wide variety of eventualities, together with hardware screw ups, cyberattacks, facts corruption, natural failures, and extra.

Stakeholder Analysis:

Key stakeholders involved in this project include IT administrators, business continuity managers, and end-users. Their roles and responsibilities will be crucial throughout the project.

Recovery Objectives:

Our primary recovery objectives are as follows:

- Recovery Point Objective (RPO): For critical databases, RPO is set at 15 minutes, meaning no more than 15 minutes of data can be lost.

- Recovery Time Objective (RTO): RTO for core services is set at 2 hours, ensuring systems are operational within this timeframe.

Risk Assessment:

We have identified potential risks, such as server failures, data corruption, and regional natural disasters. Understanding these risks allows us to better plan for disaster scenarios.

High-Level Plan:

Our initial strategy is to leverage IBM Cloud Virtual Servers to create a secondary site in a geographically distinct region. This will facilitate data replication, automatic failover, and rapid recovery.

**Phase 2: Innovation**

IBM cloud is a packaged software offering which is used to setup a private cloud on the IaaS of the users choosing. Here we focus on a Disaster Recovery use case where ICP is used to setup a Kubernetes based Private cloud on VMWare as described in ICP backup.

Innovative Technologies:

We will leverage cutting-edge technologies, such as cloud computing, virtualization, and automation, to revolutionise our disaster recovery solution. These technologies will provide scalability and cost-efficiency.

IBM Cloud Services:

IBM Cloud Virtual Servers will serve as the foundation of our disaster recovery solution. Their flexibility and high availability options make them an ideal choice for ensuring system availability.

Scalability:

Our solution will be designed to scale dynamically, ensuring it can adapt to changing workloads and demands without manual intervention.

Cost Analysis:

A preliminary cost analysis reveals that our chosen solution can result in significant cost savings compared to traditional disaster recovery methods. It minimises the need for physical infrastructure and provides a pay-as-you-go pricing model.

Compliance and Security:

Our solution complies with industry regulations, and robust security measures will protect sensitive data. Encryption and access controls will be implemented at all levels.

Integration:

The disaster recovery solution will seamlessly integrate with our existing infrastructure and applications, ensuring that business operations continue without interruption.

**Phase 3: Development**

After completing our major ideas and plans on phase 2 , here we are at phase 3 , elaborating our project with detailed explanation and building , through an completely evaluated documentation of the given phase

Developing a disaster recovery (DR) plan for a project using IBM Cloud virtual servers involves several key steps to ensure the continuity of your business operations in the event of a disaster or service interruption. Here's a step by step guide to create a robust disaster recovery plan for IBM Cloud virtual server environment

1. Assessment and Requirements Gathering

2. Backup and Replication

3. Failover Planning

4. Network and Connectivity

5. Testing and Validation

6. Monitoring and Alerting

7. Documentation and Communication

Technical Architecture:

The disaster recovery solution's technical architecture consists of primary and secondary sites. Each site hosts IBM Cloud Virtual Servers, with network configurations and server specifications optimised for rapid recovery.

Configuration Steps:

The deployment process involves creating and configuring virtual servers in the primary and secondary sites. Data replication and backup procedures will be implemented using IBM Cloud Object Storage.

Automation:

Automation scripts and tools will be used to streamline the setup and maintenance of the disaster recovery solution. This minimises the potential for human error and ensures efficiency.

Failover Plan:

In the event of a disaster, a well-documented failover plan outlines the process of transitioning from the primary environment to the secondary environment. This includes DNS updates and automatic system recovery procedures.

Monitoring and Alerts:

We will implement monitoring tools to continuously assess the health of our systems. Alerts will be configured to notify the relevant teams immediately in case of any issues.

**Phase 4: Testing and Replication**

Testing - Failover Test (IBM Cloud Virtual Servers)

Testing your failover process is a critical phase in validating the readiness of your disaster recovery plan with IBM Cloud virtual servers. This test ensures that your website can swiftly and effectively transition from the primary virtual server to the backup virtual server while adhering to your recovery time objectives (RTO).

1. Select a Test Window

2. Backup Data

3. Initiate Failover

4. IBM Cloud Virtual Server Redirection

5. Monitor and Document

6. Functional Testing

7. Performance Evaluation

8. Rollback (Optional)

9. Analysis and Optimization

Testing Strategy:

Our testing strategy includes regular failover tests, data consistency checks, and other forms of validation. Testing will be conducted on a predetermined schedule to ensure readiness.

Test Results:

During testing, we observed a successful failover process with minimal downtime and data loss. A comprehensive test report outlines the results, any issues encountered, and improvements made.

Data Replication:

Data is replicated to the secondary site in near real-time. Regular synchronisation ensures that critical data is always up-to-date and readily accessible in the event of a disaster.

CONFIGURING REPLICATION

Configuring replication for a disaster recovery plan using IBM Cloud virtual servers involves ensuring that data and configurations are synchronised between the primary and backup servers. To achieve this, you can use various methods and services, depending on your specific requirements and the services available in IBM Cloud. Here's a high-level overview of the process:

1. Data Replication

2. File System Replication

3. Backup and Restore

4. Database Backup and Restore

5. Synchronisation Tools

6. Content Delivery Networks (CDNs)

7. Testing and Validation

8. Monitoring and Alerting

9. Failover and Failback Procedures

10. Documentation

Let us look into a basic python code which explains our process

import requests

PRIMARY\_SERVER = "http://primary-server"

SECONDARY\_SERVER = "http://secondary-server"

def check\_primary\_health():

try:

response = requests.get(PRIMARY\_SERVER)

return response.status\_code == 200

except Exception as e:

return False

def initiate\_failover():

try:

response = requests.post(SECONDARY\_SERVER + "/failover")

return response.status\_code == 200

except Exception as e:

return False

if not check\_primary\_health():

if initiate\_failover():

print("Failover initiated successfully")

else:

print("Failover failed")

else:

print("Primary server is healthy")

This is a basic script for checking the health of a primary server and initiating a failover to a secondary server if the primary server is deemed unhealthy. Here's an overview of how the script works:

1. Importing the `requests` library:

- The code starts by importing the `requests` library, which is commonly used for making HTTP requests. This library allows the code to send HTTP GET and POST requests to interact with servers.

2. Defining Constants:

- `PRIMARY\_SERVER` and `SECONDARY\_SERVER` are defined as constants with the URLs of the primary and secondary servers. These URLs specify the locations where the code will check the health and initiate a failover.

3. `check\_primary\_health` function:

- Purpose: This function is responsible for checking the health of the primary server.

- Details:

a. It uses a `try...except` block to handle potential exceptions that might occur during the HTTP request.

b. Within the `try` block, it sends an HTTP GET request to the `PRIMARY\_SERVER` using `requests.get(PRIMARY\_SERVER)`.

c. It checks the response status code (`response.status\_code`) to see if it's equal to 200, which is a typical HTTP status code for a successful response.

d. If the status code is 200, the function returns `True`, indicating that the primary server is healthy.

e. If there's an exception (e.g., network issues or if the primary server is unreachable), the function returns `False` to indicate that the primary server is not healthy.

4. `initiate\_failover` function:

- Purpose: This function is responsible for initiating a failover to the secondary server.

- Details:

a. Like the `check\_primary\_health` function, it also uses a `try...except` block to handle exceptions during the HTTP request.

b. Within the `try` block, it sends an HTTP POST request to the `SECONDARY\_SERVER + "/failover"` URL using `requests.post(SECONDARY\_SERVER + "/failover")`.

c. It checks the response status code to see if it's equal to 200, indicating a successful failover initiation.

d. If the status code is 200, the function returns `True`, signifying that the failover initiation was successful.

e. If there's an exception during the request, it returns `False` to indicate that the failover initiation failed.

5. Main Part of the Code:

- The main part of the code is outside of the functions and executes the following steps:

a. It calls the `check\_primary\_health` function to check the health of the primary server.

b. If the primary server is not healthy (i.e., the function returns `False`), it proceeds to initiate a failover by calling the `initiate\_failover` function.

c. If the failover initiation is successful, it prints "Failover initiated successfully." If the initiation fails, it prints "Failover failed."

d. If the primary server is healthy (i.e., the `check\_primary\_health` function returns `True`), it prints "Primary server is healthy."

This code represents a basic failover mechanism, often used in high-availability systems. It continuously monitors the primary server and switches to the secondary server if the primary server becomes unhealthy, ensuring uninterrupted service. The `requests` library enables communication with these servers over HTTP.

Documentation Updates:

It's essential to maintain up-to-date documentation. Any changes or updates to the disaster recovery solution will be documented and communicated to the relevant stakeholders.

**Conclusion:**

In conclusion, the implementation of a robust disaster recovery solution using IBM Cloud Virtual Servers represents a pivotal step in safeguarding the continuity and resilience of our organisation's critical systems and data. This documentation has taken you through the four essential phases of our disaster recovery project: Design Thinking, Innovation, Development, and Testing and Replication. Through careful consideration and design thinking, we established the imperative need for a disaster recovery plan and set clear recovery objectives, ensuring that our systems and data remain accessible even in the face of unforeseen disruptions. Stakeholder analysis helped define roles and responsibilities, laying the groundwork for successful collaboration.

The innovative approach of utilising IBM Cloud Virtual Servers and other advanced technologies has enabled us to create a flexible and cost-efficient solution that can scale to meet our evolving needs. Compliance and security measures ensure that sensitive data is protected, and integration with our existing infrastructure ensures a seamless transition to the secondary site.

The development phase provides a detailed technical architecture, configuration steps, and automation processes that streamline setup and maintenance. The failover plan and robust monitoring and alert systems contribute to the reliability of our disaster recovery strategy.

Testing and replication are integral to the success of our plan, enabling us to validate its readiness and ensure data consistency. Regular synchronisation of data between primary and secondary sites minimises data loss in the event of a disaster.

As we move forward, it's imperative to continuously update our documentation to reflect changes in technology and business requirements. Disaster recovery is an ongoing process, and we must remain vigilant to maintain its effectiveness.

In conclusion, this disaster recovery project represents our commitment to business continuity, resilience, and the protection of our organisation's most valuable assets. We're confident that our chosen approach, leveraging IBM Cloud Virtual Servers and innovative technologies, will serve as a reliable foundation to uphold our commitment to availability and data integrity during unexpected disruptions. The meticulous planning, innovation, and testing outlined in this documentation provide the framework for a strong and agile disaster recovery solution, ensuring our organisation remains secure and operational in the face of adversity.