SRPCE

2021-2025

NAME - R.Kaviya

DEPARTMENT-CSE

ROLL-NO - 14

SUBJECT - cloud Application Development

REG-NO - 422021104014

SEMESTER- 5

IBM Cloud Disaster Recovery

Explain about disaster recovery with IBM cloud virtual servers with sample program

Disaster recovery (DR) with IBM Cloud Virtual Servers involves setting up a plan and infrastructure to ensure the availability and resilience of your applications and data in the event of a disaster or unexpected downtime. It typically involves replicating data and configurations to a secondary location, so if the primary site fails, you can quickly switch to the secondary site to minimize downtime.

Here's an overview of how you might approach disaster recovery with IBM Cloud Virtual Servers, and a simplified example program in Python for automating the failover process:

1. **Set up Redundant Virtual Servers**: Deploy your virtual servers in at least two geographically separate data centers. This ensures that if one data center experiences a disaster, the other can take over.
2. **Data Replication**: Implement data replication mechanisms to keep data synchronized between the primary and secondary sites. You can use technologies like IBM Cloud Object Storage, databases with replication features, or third-party solutions.
3. **Monitoring and Detection**: Continuously monitor the health of your primary site. Set up automated alerting to detect anomalies, such as server failures, network issues, or other problems.
4. **Failover Automation**: When a disaster is detected, automatically trigger a failover process to switch to the secondary site. Here's a simple Python program as an example:

python Copy code

```python
import requests

# Define the URL to check the primary server's health
primary_server_url = "http://primary-server.example.com/health"
```

```python
# Define the URL to promote the secondary server
secondary_server_url = "http://secondary-server.example.com/promo

def check_primary_server_health():
    try:
        response = requests.get(primary_server_url)
        if response.status_code != 200:
            return False
        return True
    except:
        return False

def initiate_failover():
    try:
        response = requests.post(secondary_server_url)
        if response.status_code == 200:
            return True
        return False
    except:
        return False

if not check_primary_server_health():
    if initiate_failover():
        print("Failover initiated successfully.")
    else:
        print("Failover failed.")
else:
    print("Primary server is healthy.")
```

In this example, we have a script that periodically checks the health of the primary server. If it detects a failure, it initiates a failover by sending a request to the secondary server to promote it.

Please note that this is a simplified example. In a real-world scenario, you would likely need more advanced automation, load balancing, and data synchronization strategies. Additionally, IBM Cloud provides services like IBM Resiliency Orchestration, which can help streamline and automate disaster recovery processes.