Disaster Recovery with IBM Cloud Virtual servers

ARUNKUMAR A (420721104004)

CODE:

```
import ibm_boto3
import ibm_botocore
import time

# Define your IBM Cloud credentials
ibm_api_key = 'YOUR_API_KEY'
ibm_service_instance_id = 'YOUR_SERVICE_INSTANCE_ID'
ibm_auth_endpoint = 'https://iam.cloud.ibm.com/identity/token'
ibm_endpoint = 'https://cloud.ibm.com'

# Define your virtual server details
primary_vsi_name = 'primary-vsi'
backup_vsi_name = 'backup-vsi'
datacenter = 'dal10'  # Choose the data center region
image_id = 'r006-abc123'  # Choose the image ID for your virtual server
ssh_key = 'your-ssh-key'  # SSH key name
ssh_key_id = 'your-ssh-key-id'  # SSH key ID

# Create a virtual server client
client = ibm_boto3.client(
```

```
    'vpc',
    ibm_api_key_id=ibm_api_key,
    ibm_service_instance_id=ibm_service_instance_id,
    ibm_auth_endpoint=ibm_auth_endpoint,
    endpoint_url=ibm_endpoint,
)

# Create a function to provision a new virtual server
def create_virtual_server(vsi_name, datacenter):
    try:
        response = client.create_instance(
            instance_name=vsi_name,
            profile={'name': 'bx2-2x8'},
            keys=[ssh_key_id],
            image=image_id,
            primary_network_interface={'name': 'eth0'},
            zone=datacenter,
        )
        return response['instance']['id']
    except ibm_botocore.exceptions.ClientError as e:
        print(f"Error creating virtual server {vsi_name}: {str(e)}")

# Create a function to replicate data from primary to backup server
(simplified)
```

```
def replicate_data(primary_vsi, backup_vsi):
    print("Replicating data from primary to backup server...")

    # Implement your data replication logic here (e.g., rsync, database replication)

    print("Data replication completed successfully.")

# Main disaster recovery function
def disaster_recovery():
    # Create a primary virtual server
    primary_vsi_id = create_virtual_server(primary_vsi_name, datacenter)
    print(f"Provisioning primary virtual server ({primary_vsi_id}).")

    # Create a backup virtual server
    backup_vsi_id = create_virtual_server(backup_vsi_name, datacenter)
    print(f"Provisioning backup virtual server ({backup_vsi_id}).")

    # Wait for both virtual servers to be provisioned
    while True:
        primary_status = client.get_instance(instance_id=primary_vsi_id)['instance']['status']
```

```
        backup_status =
client.get_instance(instance_id=backup_vsi_id)['instance']['status']

        if primary_status == 'running' and backup_status == 'running':
            break
        time.sleep(30)

    print("Both virtual servers are running.")

    # Replicate data from primary to backup server
    replicate_data(primary_vsi_id, backup_vsi_id)

    print("Disaster recovery completed successfully.")

if _name_ == "_main_":
    disaster_recovery()
```

OUTPUT:

The script will display messages such as:

```
Provisioning primary virtual server ({primary_vsi_id}).
Provisioning backup virtual server ({backup_vsi_id}).
```

These messages indicate that the primary and backup virtual servers are being provisioned.

The script will periodically check the status of the virtual servers and display messages like:

```
Both virtual servers are running.
```

This message indicates that both the primary and backup virtual servers have started running.

When the data replication process is triggered, you will see:

```
Replicating data from primary to backup server...
```

After the data replication (or any other specific disaster recovery process you implement) is completed, you will see:

```
Data replication completed successfully.
```

Finally, the script will display:

```
Disaster recovery completed successfully.
```