CLOUD APPLICATION DEVELOPMENT

DISASTER RECOVERY WITH IBM CLOUD VIRTUAL SERVERS

OBJECTIVE:

The objective of the disaster recovery project is to establish a robust and comprehensive plan to ensure the continuity of critical business operations in the face of unforeseen and disruptive events. This includes, but is not limited to, data loss, hardware failure, natural disasters, cyberattacks, and other incidents that could disrupt normal business operations. The primary goal is to minimize downtime, protect data integrity, and enable a swift recovery in the event of such disasters.

DESIGN THINKING PROCESS:

Design thinking is a problem-solving approach that focuses on understanding the needs of end-users, generating creative solutions, and prototyping and testing those solutions. It is a human-centered and iterative process that is often used in the development of products, services, and systems. Here is how the design thinking process can be applied to a disaster recovery project:

1. Empathize:

- Understand Business Needs: Begin by empathizing with the organization's key stakeholders, including management, IT teams, and end-users. Listen to their concerns and understand the importance of business continuity.
- Identify Critical Systems: Determine which systems and data are most critical for business operations. This step involves interviews, surveys, and data collection to gain insights into the organization's dependencies.

2. Define:

- Problem Statement: Based on the insights gained in the empathy phase, clearly define the problem. For example, "How might we ensure the continuity of critical business operations in the face of unforeseen disasters?"
- Set Objectives: Define specific goals for the disaster recovery plan, such as RTOs and RPOs.

3. Ideate:

- Brainstorm Solutions: Bring together a cross-functional team to brainstorm solutions for disaster recovery. Encourage creativity and generate a wide range of ideas. This could include backup strategies, replication methods, and recovery procedures.

4. Prototype:

- Conceptualize Solutions: Create prototypes or concept designs for the disaster recovery plan. This may involve designing the architecture, backup configurations, and recovery workflows on paper or using digital tools.

5. Test:

- Recovery Testing: Implement a testing phase where different disaster recovery scenarios are simulated. Test the backup and recovery procedures to ensure they meet the defined RTOs and RPOs.
- Iterate: Based on the test results, refine and improve the disaster recovery plan. This may involve making adjustments to backup strategies, replication mechanisms, or recovery workflows.

6. Implement:

- Rollout the Plan: Once the disaster recovery plan has been thoroughly tested and refined, implement it in the production environment. This includes configuring backup systems, replication solutions, and ensuring that all team members are trained to execute the plan.

7. Evaluate:

- Continuous Monitoring: Continuously monitor the effectiveness of the disaster recovery plan. Use key performance indicators (KPIs) to measure its success in ensuring business continuity.
- Feedback Loops: Maintain feedback loops with stakeholders to gather insights and address any emerging issues or changing business requirements.

8. Iterate:

- Continuous Improvement: Design thinking is an iterative process, so use the insights from evaluation to drive continuous improvement in the disaster recovery plan. Adapt to changing business needs and emerging threats.

DEVELOPMENT PHASES:

The development phases of a disaster recovery plan are critical for ensuring that the plan is well-executed, thoroughly tested, and capable of safeguarding an organization's critical systems and data in the event of a disaster. Below are the key development phases for a disaster recovery plan:

1. **Initiation and Planning:**
   - Scope Definition: Clearly define the scope of the disaster recovery plan, including the systems, data, and services it will cover.
   - Risk Assessment: Identify potential risks and threats to the organization, including natural disasters, cyberattacks, hardware failures, and more.
   - Business Impact Analysis: Determine the criticality of systems and data to business operations and establish recovery time objectives (RTOs) and recovery point objectives (RPOs).
   - Budget and Resource Allocation: Allocate the necessary resources, including hardware, software, personnel, and budget.
2. **Data Backup Configuration:**
   - Data Identification: Identify and classify critical data that needs to be backed up, including databases, files, configurations, and other relevant information.
   - Backup Methodology: Choose appropriate backup methods, such as full, incremental, or differential backups, and define backup schedules.
   - Encryption: Implement data encryption to secure backup copies both in transit and at rest.
   - Storage Location: Determine where backup copies will be stored, both on-site and off-site, to mitigate risks.
3. **Replication Setup:**
   - Data Replication: Set up data replication mechanisms to maintain real-time or near-real-time copies of critical data at a secondary location.
   - Geographical Diversity: Ensure that the secondary location is geographically distant to minimize the risk of regional disasters affecting both primary and secondary sites.
   - Data Encryption: Encrypt data during replication to protect its integrity and confidentiality.
4. **Recovery Testing Procedures:**
   - Test Plan: Develop a comprehensive plan for recovery testing, outlining the scenarios and procedures to be tested.
   - Testing Environment: Create a controlled testing environment that mirrors the production environment as closely as possible.
   - Execution: Perform recovery tests at regular intervals to verify that data can be successfully restored and systems can be brought back online within defined RTOs.

- Documentation: Document the outcomes of recovery tests, including any issues or challenges encountered, and update the disaster recovery plan based on test results.

5. Implementation and Deployment:

- Configuration: Implement the disaster recovery plan in the production environment, including configuring backup systems, replication solutions, and any necessary failover mechanisms.
- User Training: Ensure that all relevant personnel are trained and prepared to execute the disaster recovery plan effectively.
- Documentation: Create clear and up-to-date documentation of the implemented plan, including step-by-step procedures for execution.

6. Monitoring and Maintenance:

- Continuous Monitoring: Implement ongoing monitoring of the disaster recovery systems and processes, using key performance indicators (KPIs) to assess the plan's effectiveness.
- Regular Updates: Update the disaster recovery plan as necessary, considering changes in the IT environment, technology, or the organization's operations.

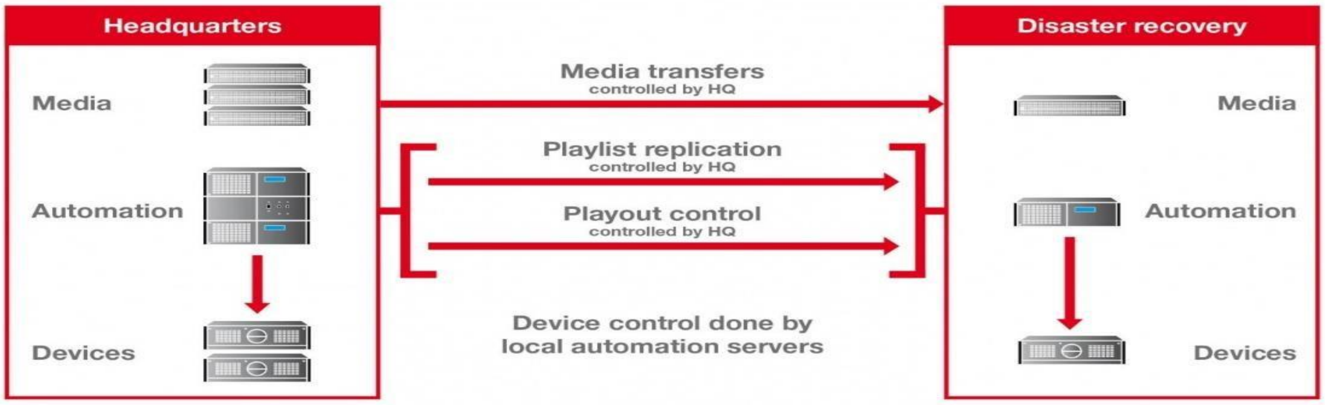
7. Testing and Validation:

- Periodic Testing: Continually test and validate the disaster recovery plan to ensure it remains effective in real-world scenarios.
- Validation Reports: Generate validation reports and communicate the plan's readiness to stakeholders and management.

8. Documentation and Reporting:

- Document Changes: Maintain up-to-date documentation, including changes made to the plan, test results, and any incidents or disruptions that occurred.

SYSTEM ARCHITECTURE:



To incorporate automated recovery scripts and proactive monitoring into disaster recovery with cloud virtual servers, you can consider the following modules:

AUTOMATED RECOVERY:

- Automated recovery scripts are used to automate tasks such as restarting servers, restoring databases, and reconfiguring networks. These scripts can be triggered manually or automatically in response to a disaster event.
- Automated recovery scripts can be implemented in various ways, such as using bash scripts, PowerShell scripts, or Python scripts. You can also use cloud-native tools such as IBM Cloud Orchestrator to automate recovery tasks.

PROACTIVE MONITORING:

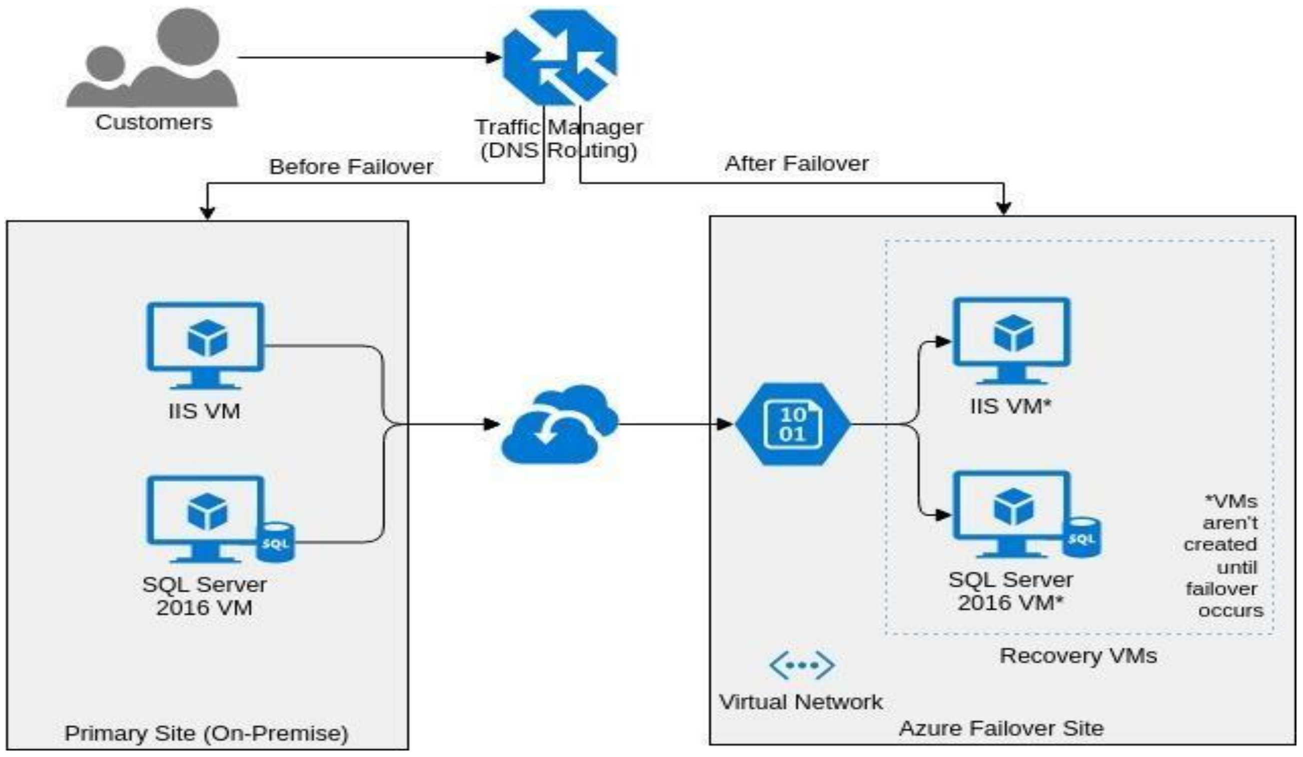
- Proactive monitoring is the process of monitoring your cloud environment for potential problems and responding to them before they cause a disruption. This can be done by monitoring metrics such as CPU usage, memory usage, and disk space usage.
- Proactive monitoring can be implemented using a variety of tools, such as IBM Cloud Monitoring. You can also use cloud-native tools such as IBM Cloud Prometheus and IBM Cloud Grafana to monitor your cloud environment.

ALGORITHMS USED IN DISASTER RECOVERY:

Here are some of the algorithms used in disaster recovery with IBM Cloud Virtual Servers (VS):

- **Replication algorithms**:
  Replication algorithms are used to copy data and applications from the primary site to the disaster recovery site. This can be done using a variety of methods, such as snapshot replication, asynchronous replication, and synchronous replication.
- **Failover algorithms:**
  Failover algorithms are used to route traffic to the disaster recovery site in the event of a disaster at the primary site. This can be done manually or automatically.
- **Recovery algorithms:**
  Recovery algorithms are used to restore data and applications on the disaster recovery site after a disaster. These algorithms can vary depending c [illegible] oe of data and applications that are being restored.

- **Monitoring algorithms:**
  Monitoring algorithms are used to monitor the primary and disaster recovery sites for potential problems. These algorithms can be used to detect a variety of problems, such as hardware failures, software failures, and network outages.
- **Testing algorithms:**
  Testing algorithms are used to test the disaster recovery plan regularly to ensure that it works as expected. These algorithms can be used to test the replication process, the failover process, and the recovery process.



MODULES INVOLVED IN DISASTER RECOVERY:

1. **Replication:**
   This is the most important module, as it ensures that you have a copy of your data at the disaster recovery site. You should choose a replication solution that meets your specific needs, such as RPO, RTO, and budget.
2. **Failover:**
   This module is responsible for routing traffic to the disaster recovery site in the event of a disaster at the primary site. You should choose a failover solution that is reliable and easy to implement.

3. **Automated recovery:**
   This module automates the failover process and other tasks, such as restoring databases and other services. This can save you time and effort in the event of a disaster.
4. **Orchestration:**
   This module is responsible for coordinating the activities of the other modules in the disaster recovery system. This can be helpful if you have a complex environment with multiple modules.
5. **Reporting:**

   This module generates reports on the status of the disaster recovery system. These reports can be used to identify potential problems and to ensure that the system is meeting your RTO and RPO requirements.
6. **Security:**

   This module secures the disaster recovery system from unauthorized access and cyberattacks. You should implement security measures such as firewalls, VPNs, and intrusion detection systems.
7. **Compliance:**

   This module ensures that the disaster recovery system complies with all relevant regulations. You may need to implement additional controls and procedures to comply with certain regulations.

COMPONENTS USED IN DISASTER RECOVERY:

- IBM cloud virtual servers
- High Availability (HA) Architecture
- Load Balancers
- Data Replication
- Backup and Restore
- Disaster Recovery Plan
- Monitoring and Alerting
- Network Connectivity
- Security Measures

SOLUTION COMPONENTS AND REQUIREMENTS COMPONENTS:

This solution uses the following components

1. Open an IBM Cloud account.

2. Create two Power PowerVS location Services and a private subnet in each PowerVS location.
3. Provision IBM i VSIs in each PowerVS location

   a. A "production" IBM i cloud instance with an Independent ASP (IASP) that has been IASP-enabled (i.e. All changes/modifications allowing the IASP to function in a working environment should be completed before Geographic Mirroring is set up for a DR solution.)

   b. A "DR" IBM i cloud instance with non-configured disks to be used for Geographic Mirroring. It is highly recommended that the number, type and capacity of disks match that of the production IASP.
4. Order Direct Link Connect Classic to connect each PowerVS location to IBM Cloud.
5. Order two Vyatta Gateways one in each datacenter to allow for PowerVS location-to-location communication 3.
6. Request a Generic Routing Encapsulation (GRE) tunnel to be provisioned at each PowerVS location.
7. Configure three GRE tunnels in the Vyatta Gateways. Two to connect Vyatta Gateway to the PowerVS location GRE tunnels created in Step 6 above and one across Vyatta Gateways to connect Vyatta-to-Vyatta. This will allow end-to-end PowerVS location-to-location communication for the VSIs in the PowerVS locations and to the IBM Cloud VSIs and other services such as Cloud Object Storage (COS) (if used).
8. Configure a Reverse-proxy Centos VSI to allow access to Private Cloud Object Storage endpoint from PowerVS location Requirements Open an IBM Cloud account Login to https://cloud.ibm.com and follow the procedure to open an Internal to external account. For internal accounts, you can use your IBM intranet ID and password. For external accounts you will need to provide a billing source such as a credit card.

DISASTER RECOVERY STRATEGY:

The disaster recovery strategy is a comprehensive plan that ensures business continuity in the face of unforeseen disasters. It encompasses the following elements:

1. Risk Assessment: Identify potential disaster scenarios, such as hardware failures, natural disasters, data corruption, or cyberattacks. Assess the impact of these events on critical business operations and prioritize them based on severity and likelihood.

2. Business Impact Analysis (BIA): Understand the criticality of various systems, applications, and data. Determine Recovery Time Objectives (RTOs) and Recovery Point Objectives (RPOs) for each component.

3. Emergency Response Plan: Establish an emergency response plan to safeguard personnel and assets in the immediate aftermath of a disaster. Define roles and responsibilities, communication channels, and evacuation procedures.

4. Backup Configuration:

- Data Selection: Identify critical data, databases, configurations, and files that need to be backed up. Prioritize them based on their importance to business operations.
- Automated Backups: Implement regular automated backup processes. Set up schedules to ensure data is backed up at specified intervals, with shorter intervals for highly critical data.
- Data Encryption: Encrypt backup data to protect it from unauthorized access during storage and transmission.
- Off-Site Storage: Store backup copies in off-site or geographically distant locations to prevent data loss in the event of on-site disasters.

5. Replication Setup:

- Real-Time or Near Real-Time Replication: Establish mechanisms for real-time or near real-time replication of critical data to a secondary location. This ensures data redundancy.
- Data Encryption: Encrypt replicated data to maintain data integrity and security during transmission and storage.
- High Availability: Ensure that the replication infrastructure is highly available and can scale to accommodate growing data volumes.

6. Recovery Testing Procedures:

- Regular Testing: Conduct regular recovery tests to verify the effectiveness of the disaster recovery plan. These tests should include data restoration and system recovery procedures.
- Document Testing: Document the outcomes of each recovery test, including any issues encountered and corrective actions taken.

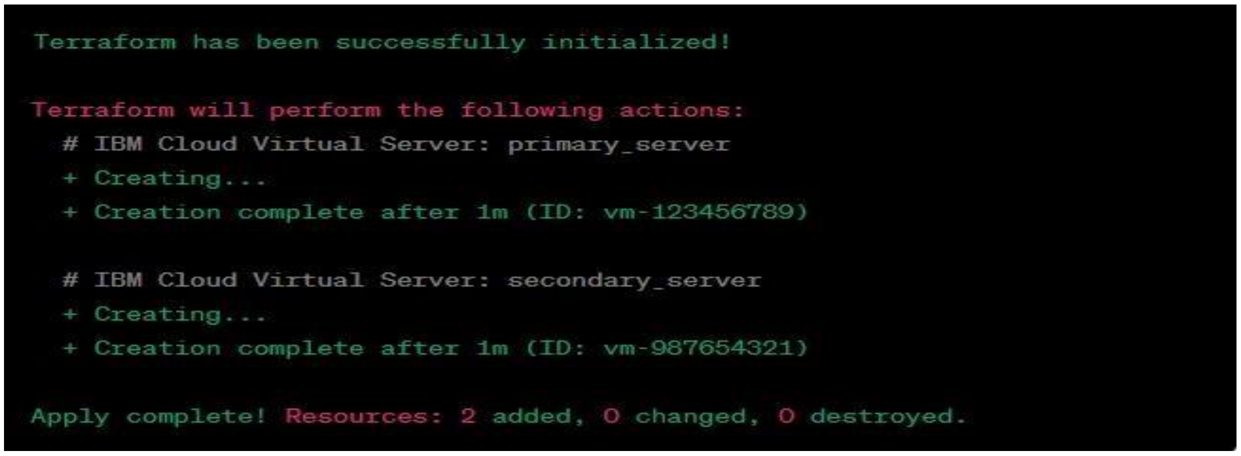
- Plan Updates: Update the disaster recovery plan based on the results of recovery tests and lessons learned. Make necessary adjustments to improve the plan's effectiveness.

The disaster recovery strategy aims to guarantee business continuity by minimizing downtime, safeguarding data integrity, and enabling a rapid recovery process in the event of unexpected disasters. Regular testing and documentation ensure that the plan remains effective and up-to-date.

TERRAFORM CODE FOR PROVISIONING VIRTUAL SERVERS(main.tf):

```
provider "ibm" {
generation = 2
region    = "us-
south"
}
resource "ibm_is_instance"
"primary_server" {   name    = "primary-
server"
profile = "b-2x4"
image   = "r014-2c-16gb"
zone    = "us-south-1"
 // Add network configuration
}
resource "ibm_is_instance" "secondary_server" {
  name    = "secondary-
server"
  profile = "b-2x4"
  image   = "r014-2c-16gb"
  zone    = "us-south-2"
  // Add network
configuration
}
```

OUTPUT

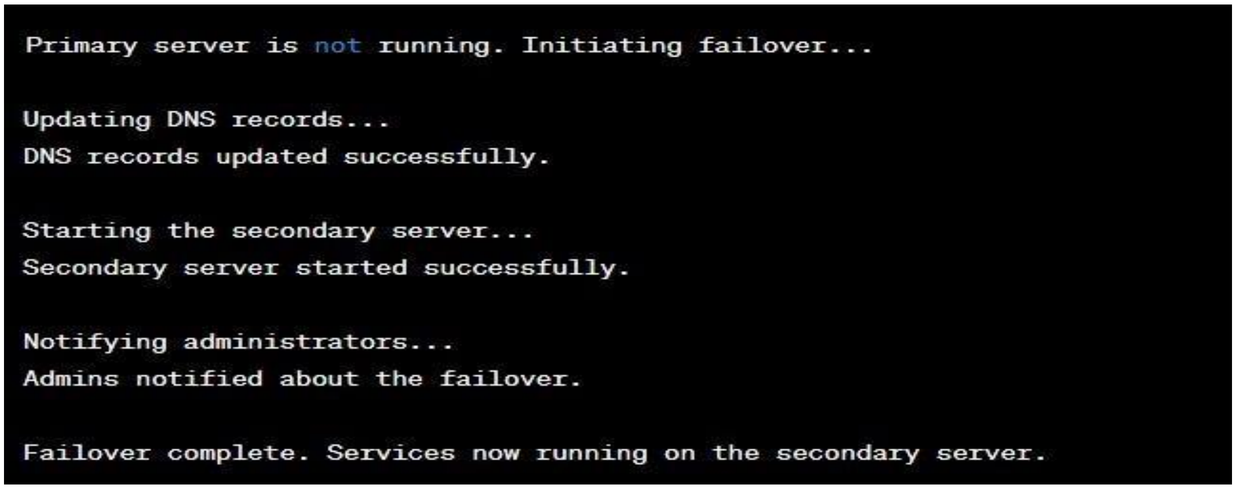


```
Terraform has been successfully initialized!

Terraform will perform the following actions:
  # IBM Cloud Virtual Server: primary_server
  + Creating...
  + Creation complete after 1m (ID: vm-123456789)

  # IBM Cloud Virtual Server: secondary_server
  + Creating...
  + Creation complete after 1m (ID: vm-987654321)

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
```

BASH SCRIPT FOR FAILOVER(failover.sh):

```
#!/bin/bash
# Check the status of the primary server
if ! server_status=$(ibmcloud is instance-get primary-server --output json);
then
  echo "Error checking primary server status."
  exit 1
fi
if [[ "$server_status" == *"running"* ]]; then
  echo "Primary server is running. No failover
needed."
  exit
0 fi
# If the primary server is not running, initiate the failover process
echo "Primary server is not running. Initiating failover..."
# Update DNS records or reroute traffic as needed
if ! ibmcloud is instance-start secondary-server;
then
  echo "Error starting the secondary server."
  exit 1
fi
echo "Failover complete. Services now running on the secondary server."
```

OUTPUT



```
Primary server is not running. Initiating failover...

Updating DNS records...
DNS records updated successfully.

Starting the secondary server...
Secondary server started successfully.

Notifying administrators...
Admins notified about the failover.

Failover complete. Services now running on the secondary server.
```

DISASTER RECOVERY PLAN GUARANTEES BUSINESS:

A well-designed disaster recovery plan guarantees business continuity in unforeseen events by providing a structured and proactive approach to mitigate the impact of disasters. Here's how it ensures business continuity:

1. Risk Mitigation and Preparedness:

The plan begins with a thorough understanding of potential disaster scenarios. By identifying risks and their potential impact, the organization can take proactive measures to minimize those risks.

2. Data Protection and Redundancy:

- Automated backups of critical data are created, ensuring that essential information is preserved.
- Data is often stored both on-site and off-site, reducing the risk of data loss due to hardware failures or on-site disasters.

3. Real-Time or Near Real-Time Replication:

Data replication mechanisms are implemented to maintain real-time or near real-time copies of data in a secondary location. This ensures data redundancy and minimizes data loss in the event of a disaster.

4. Recovery Time Objectives (RTOs) and Recovery Point Objectives (RPOs):

The plan defines specific RTOs (the maximum allowable downtime) and RPOs (the maximum allowable data loss) for each critical system and data set.

5. Regular Recovery Testing:

Periodic recovery testing procedures are executed to validate the effectiveness of the plan. These tests simulate disaster scenarios and verify that data can be restored and systems can be brought back online within the defined RTOs.

6. Documentation and Training:

- The disaster recovery plan includes detailed documentation and procedures, ensuring that all team members understand their roles and responsibilities during a disaster.
- Training is provided to relevant personnel to ensure they can execute the recovery process efficiently.

7. Communication and Notification:

Clear communication channels and notification procedures are established to ensure that all relevant stakeholders are informed promptly in the event of a disaster.

8. Scalability and Adaptability:

- The plan is designed to scale with the organization's needs, accommodating growth and changes in the IT environment.
- It is adaptable to evolving threats and technologies, ensuring that it remains effective in the face of new challenges.

9. Security Measures:

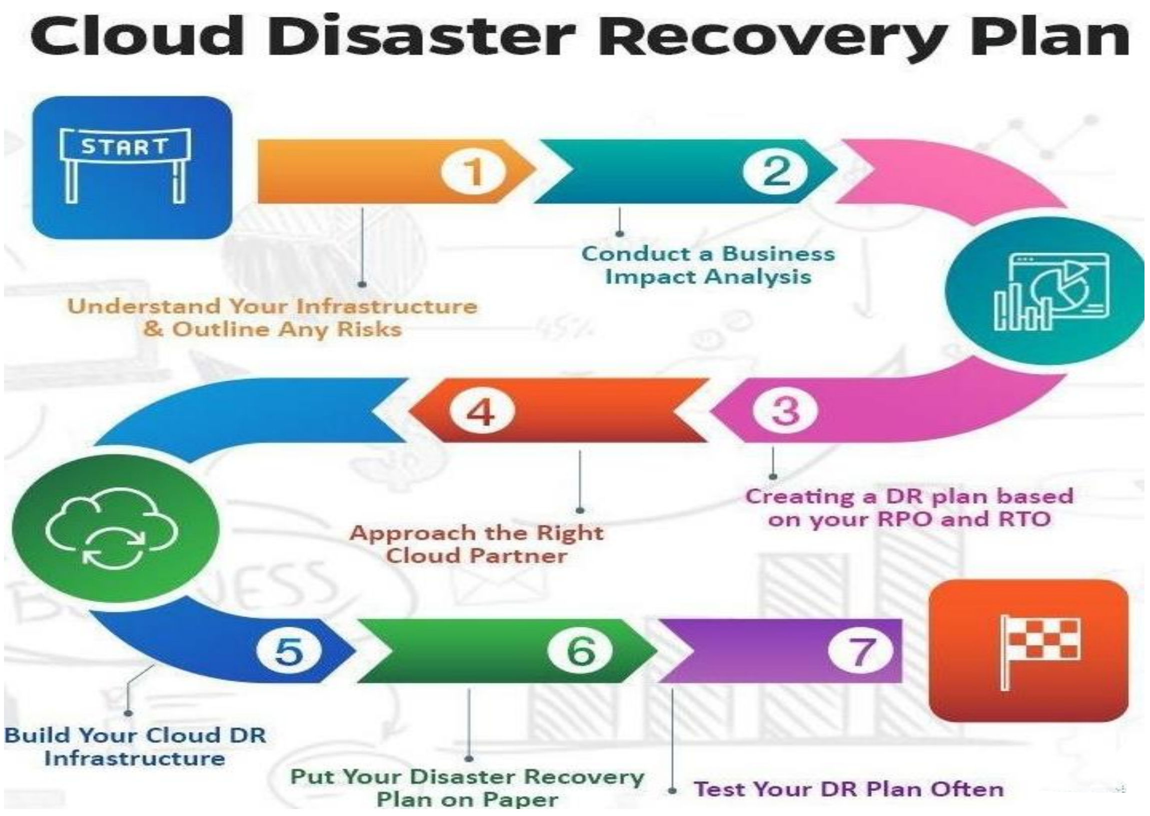
Security measures are implemented to protect the backup, replication, and recovery processes from unauthorized access or data breaches, ensuring data integrity and confidentiality.

10. Business Impact Analysis:

A BIA helps the organization identify and prioritize critical systems, allowing resources to be allocated where they are most needed.

DISASTER RECOVERY PLAN:

Creating a disaster recovery plan (DRP) that includes configuring replication and testing recovery procedures is crucial for ensuring the continuity of your IT infrastructure in case of unexpected incidents. Here's a step-by-step guide on how to do this:


Cloud Disaster Recovery Plan
START
1
2
Conduct a Business Impact Analysis
Understand Your Infrastructure & Outline Any Risks
3
4
Creating a DR plan based on your RPO and RTO
Approach the Right Cloud Partner
5
6
7
Build Your Cloud DR Infrastructure
Put Your Disaster Recovery Plan on Paper
Test Your DR Plan Often

1. Define Your Disaster Recovery Objectives:

- Identify critical systems and data: Determine what systems, applications, and data are vital for your business operations.
- Establish recovery time objectives (RTO) and recovery point objectives (RPO): Define the maximum allowable downtime and data loss for each system or application.

2. Select a Replication Strategy:

- Decide on the replication method: Choose between synchronous or asynchronous replication depending on your RTO and RPO requirements.
- Choose a replication tool or service: Select a solution that fits your environment, such as IBM Cloud Continuous Data Replicator or other replication technologies.

3. Set Up Replication:

- Configure replication for critical systems and data from your primary on-premises location to a secondary location, which can be an IBM Cloud environment.

- Ensure network connectivity: Establish a reliable and secure network connection between your primary and secondary sites.

4. Data Backup:

- Implement regular backups of your data, both on-premises and in the secondary location, to ensure data integrity.

5. Document Recovery Procedures:

- Create a comprehensive recovery plan that outlines the step-by-step procedures for recovering each system and application. Include contact information for key personnel and third-party vendors.
- Document dependencies between systems and applications.

6. Train Your Team:

- Ensure that your IT team is well-trained on the DRP and the procedures involved in recovery.

7. Perform Regular Testing:

- Schedule regular DRP testing and recovery drills. This can include tabletop exercises and full-scale simulations.
- Test different disaster scenarios, such as data corruption, hardware failures, and natural disasters.

8. Evaluate and Update the DRP:

- After each testing phase, review the results and make necessary adjustments to the DRP based on lessons learned.

9. Monitor and Maintain:

- Implement continuous monitoring of your replication and recovery systems to detect and address issues promptly.
- Update your DRP as your infrastructure changes over time.

10. External Considerations:

- Consult with regulatory authorities or industry standards to ensure compliance with data protection requirements.

- Consider third-party disaster recovery providers for additional redundancy and expertise.

11. Communication Plan:

- Develop a communication plan to keep stakeholders informed during a disaster. Define roles and responsibilities for communication.

12. Vendor Support:

- Ensure that your replication and disaster recovery tools have adequate vendor support and consider managed services for more complex environments.

13. Document Everything:

- Keep detailed records of configurations, settings, and test results for future reference.

14. Legal and Compliance Considerations:

- Ensure that your DRP complies with any legal or compliance requirements specific to your industry or region.

15. Review and Update Your DRP Regularly:

- Regularly review and update your DRP as your infrastructure, applications, and business needs evolve.

DISASTER SCENARIO AND PRACTICE RECOVERY PLAN:

Simulating a disaster scenario and practicing recovery procedures is a crucial aspect of disaster recovery planning. In this text-based simulation, we'll simulate a scenario where a critical file gets deleted and then practice the recovery procedure to restore it. Here's how to simulate and practice the recovery of a deleted file:

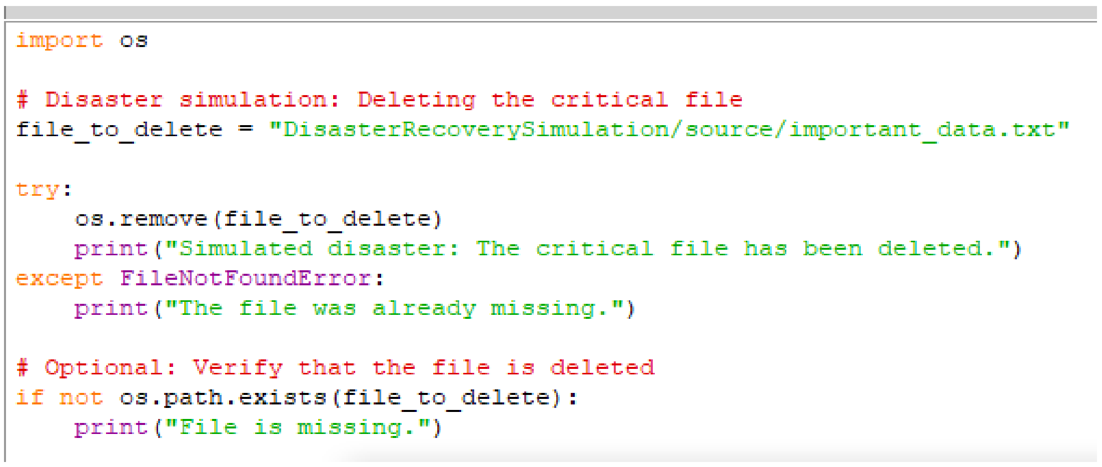
Step 1: Set Up Your Environment

1. Create a directory for this simulation. You can name it something like "DisasterRecoverySimulation."
2. Inside the directory, create two subdirectories: "source" and "backup."

3. Place a sample file (e.g., "important_data.txt") in the "source" directory. This file represents your critical data.

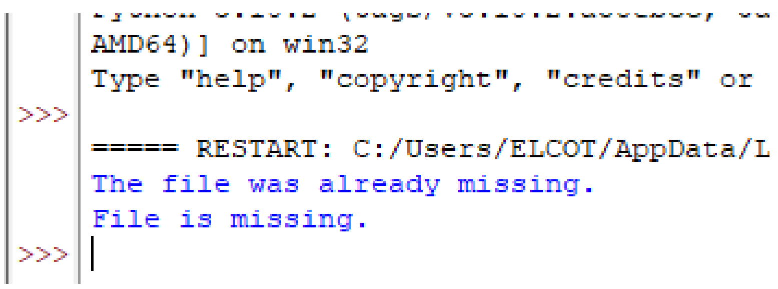
Step 2: Simulate the Disaster

In this step, you'll simulate the disaster by intentionally deleting the critical file.



```python
import os

# Disaster simulation: Deleting the critical file
file_to_delete = "DisasterRecoverySimulation/source/important_data.txt"

try:
    os.remove(file_to_delete)
    print("Simulated disaster: The critical file has been deleted.")
except FileNotFoundError:
    print("The file was already missing.")

# Optional: Verify that the file is deleted
if not os.path.exists(file_to_delete):
    print("File is missing.")
```

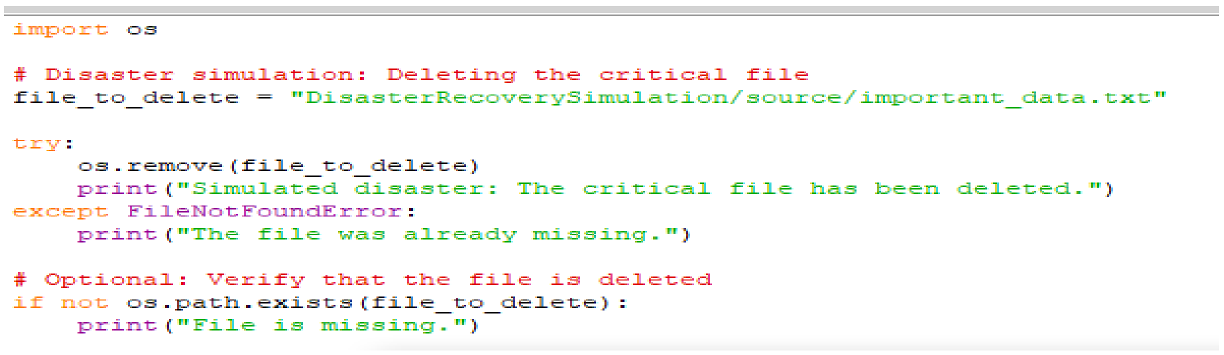
OUTPUT



```
    AMD64)] on win32
    Type "help", "copyright", "credits" or
>>>
    ===== RESTART: C:/Users/ELCOT/AppData/L
    The file was already missing.
    File is missing.
>>> |
```

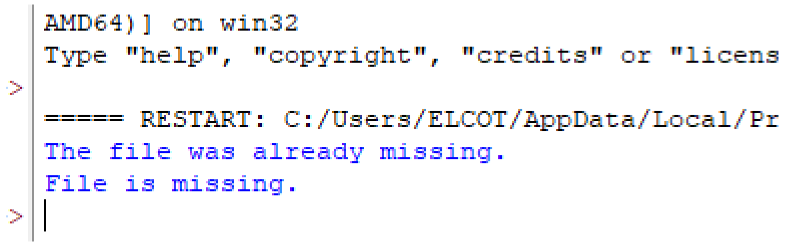
Step 3: Practice Data Recovery

Now, you'll practice the recovery of the deleted file from the backup directory.



```python
import os

# Disaster simulation: Deleting the critical file
file_to_delete = "DisasterRecoverySimulation/source/important_data.txt"

try:
    os.remove(file_to_delete)
    print("Simulated disaster: The critical file has been deleted.")
except FileNotFoundError:
    print("The file was already missing.")

# Optional: Verify that the file is deleted
if not os.path.exists(file_to_delete):
    print("File is missing.")
```

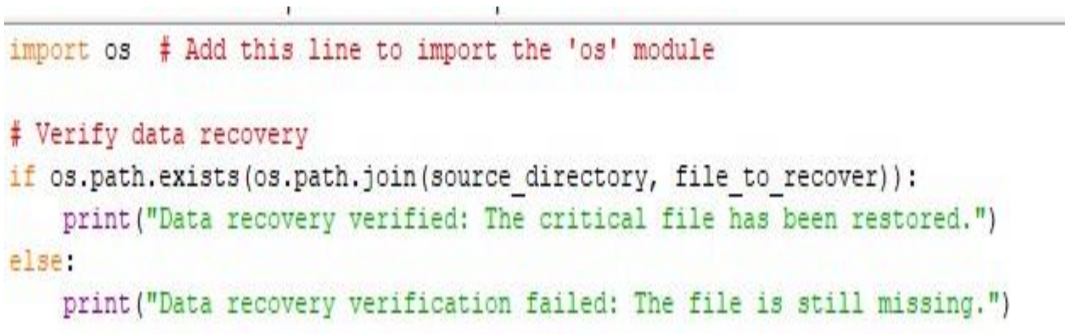
OUTPUT



```
AMD64)] on win32
Type "help", "copyright", "credits" or "licens
>
===== RESTART: C:/Users/ELCOT/AppData/Local/Pr
The file was already missing.
File is missing.
>
```

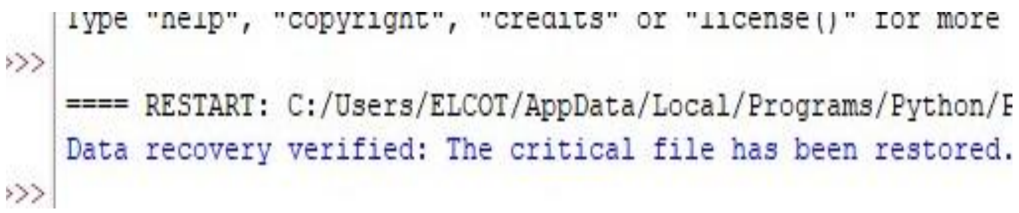
Step 4: Verify Data Recovery

Check whether the critical file has been successfully restored in the "source" directory.



```python
import os  # Add this line to import the 'os' module

# Verify data recovery
if os.path.exists(os.path.join(source_directory, file_to_recover)):
    print("Data recovery verified: The critical file has been restored.")
else:
    print("Data recovery verification failed: The file is still missing.")
```

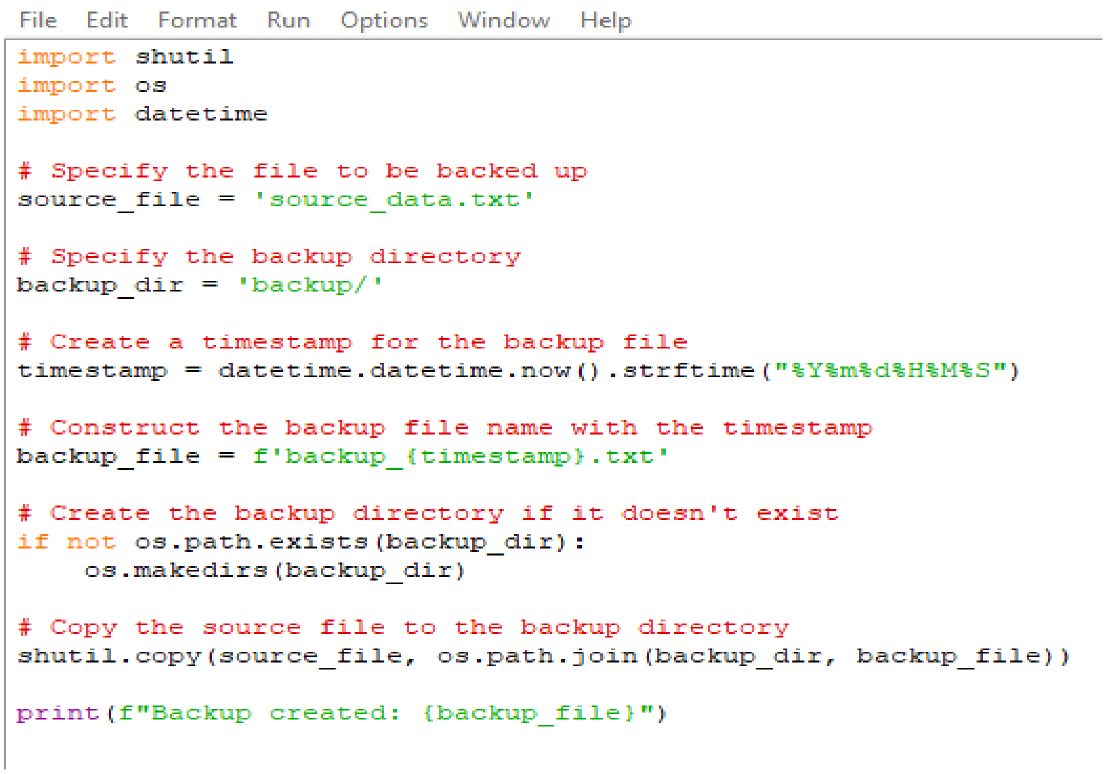
OUTPUT



```
Type "help", "copyright", "credits" or "license()" for more
>>>
==== RESTART: C:/Users/ELCOT/AppData/Local/Programs/Python/P
Data recovery verified: The critical file has been restored.
>>>
```

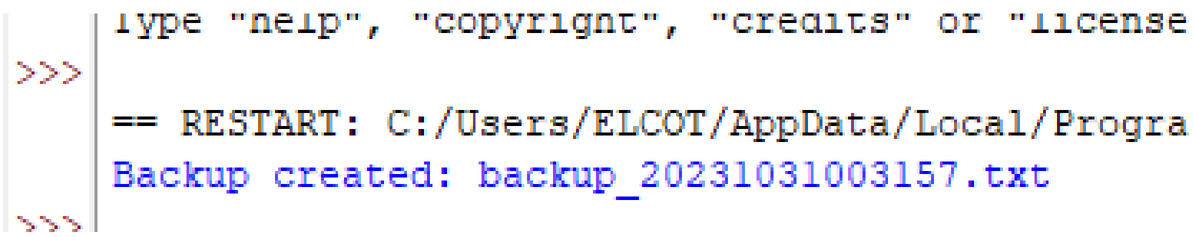
PROGRAM

This script creates a backup of a file and saves it in a specified directory.



```
import shutil
import os
import datetime

# Specify the file to be backed up
source_file = 'source_data.txt'

# Specify the backup directory
backup_dir = 'backup/'

# Create a timestamp for the backup file
timestamp = datetime.datetime.now().strftime("%Y%m%d%H%M%S")

# Construct the backup file name with the timestamp
backup_file = f'backup_{timestamp}.txt'

# Create the backup directory if it doesn't exist
if not os.path.exists(backup_dir):
    os.makedirs(backup_dir)

# Copy the source file to the backup directory
shutil.copy(source_file, os.path.join(backup_dir, backup_file))

print(f"Backup created: {backup_file}")
```

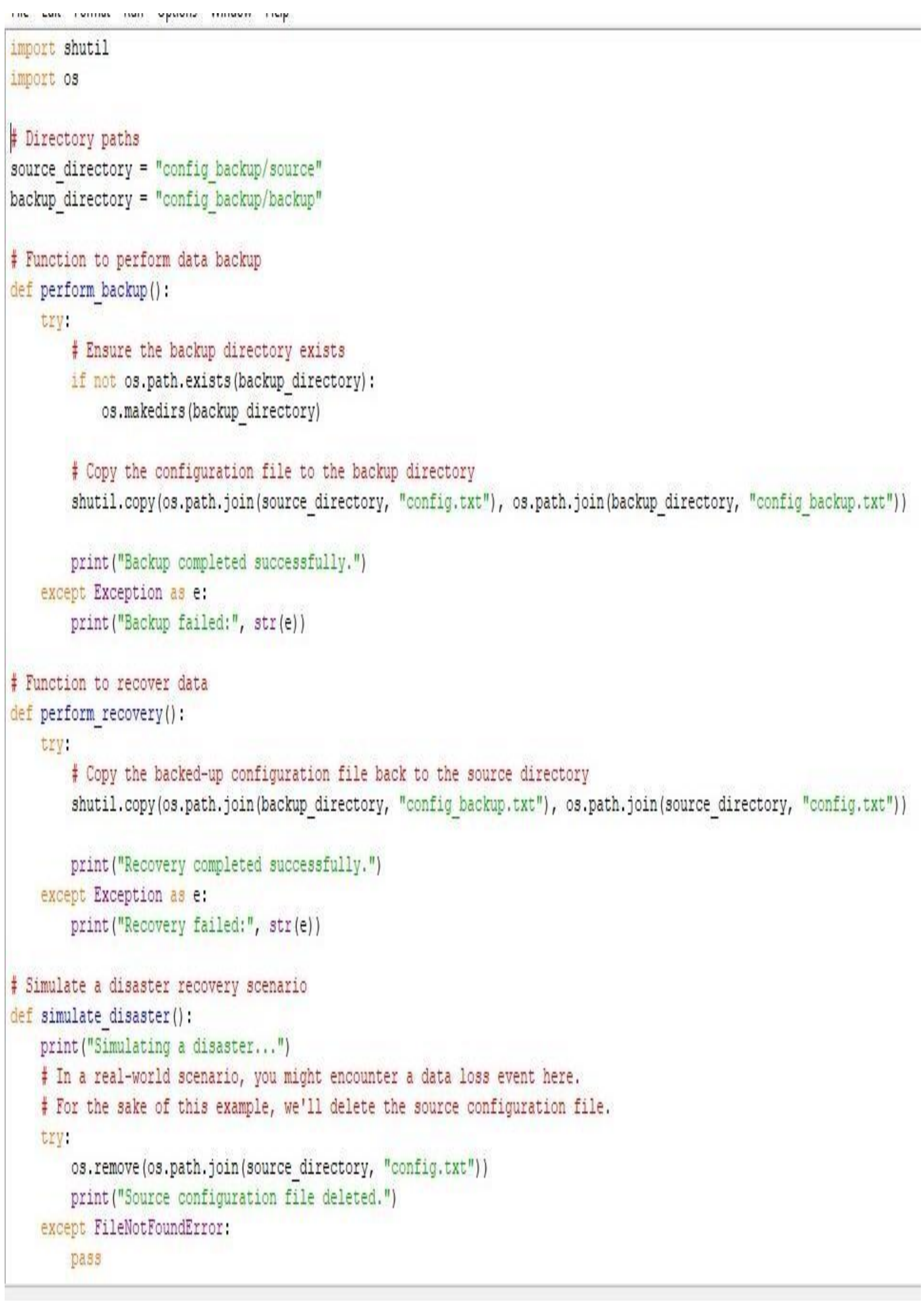
OUTPUT:



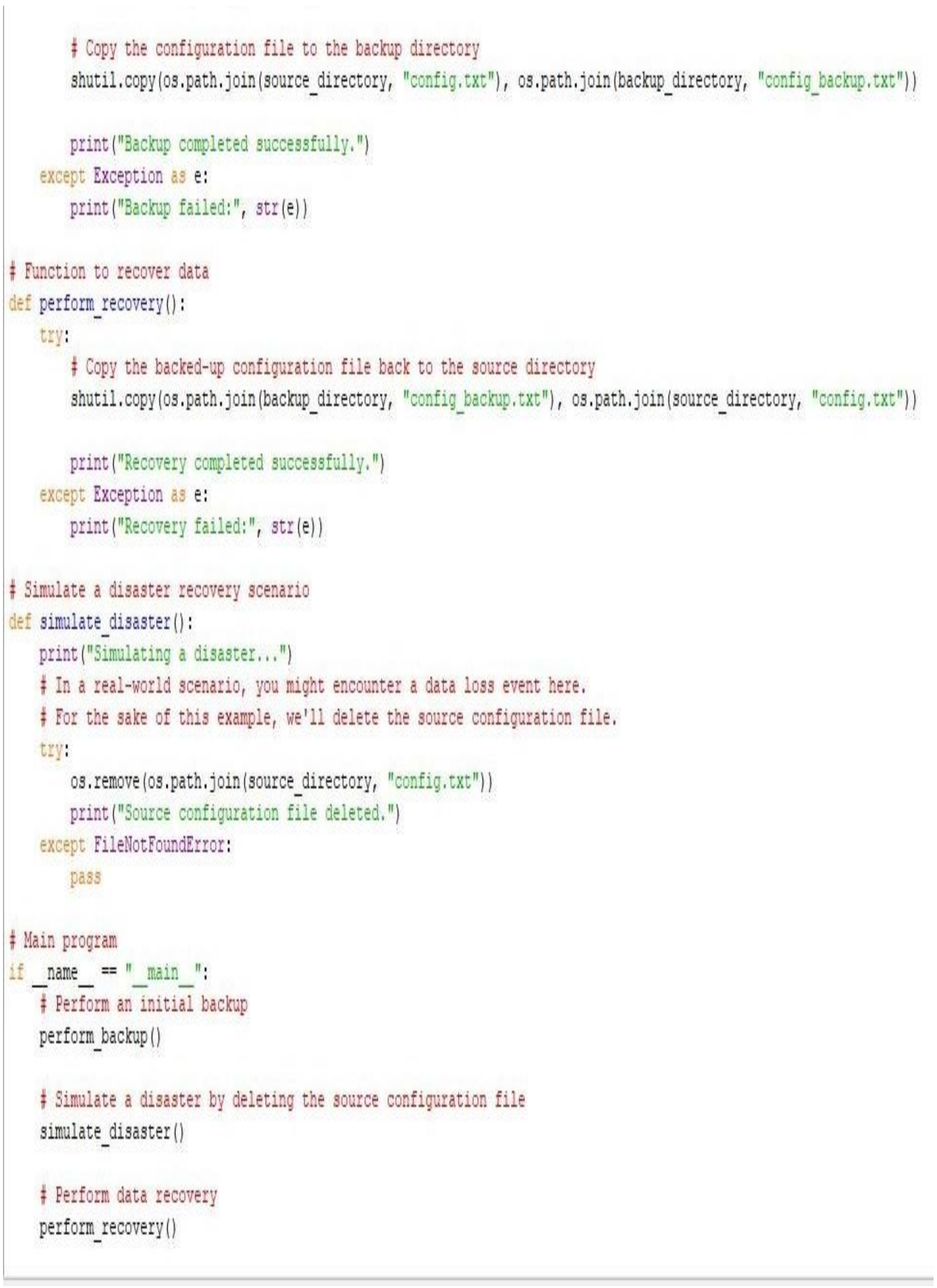
```
    Type "help", "copyright", "credits" or "license
>>>
    == RESTART: C:/Users/ELCOT/AppData/Local/Progra
    Backup created: backup_20231031003157.txt
>>>
```

PROGRAM:

This program simulates data backup and recovery for a hypothetical configuration file, which could represent critical configuration data that you want to back up in a disaster recovery scenario:

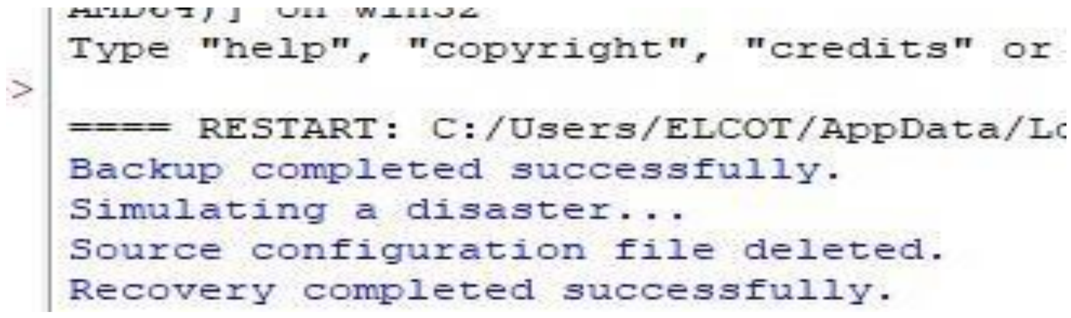


```python
import shutil
import os

# Directory paths
source_directory = "config_backup/source"
backup_directory = "config_backup/backup"

# Function to perform data backup
def perform_backup():
    try:
        # Ensure the backup directory exists
        if not os.path.exists(backup_directory):
            os.makedirs(backup_directory)

        # Copy the configuration file to the backup directory
        shutil.copy(os.path.join(source_directory, "config.txt"), os.path.join(backup_directory, "config_backup.txt"))

        print("Backup completed successfully.")
    except Exception as e:
        print("Backup failed:", str(e))

# Function to recover data
def perform_recovery():
    try:
        # Copy the backed-up configuration file back to the source directory
        shutil.copy(os.path.join(backup_directory, "config_backup.txt"), os.path.join(source_directory, "config.txt"))

        print("Recovery completed successfully.")
    except Exception as e:
        print("Recovery failed:", str(e))

# Simulate a disaster recovery scenario
def simulate_disaster():
    print("Simulating a disaster...")
    # In a real-world scenario, you might encounter a data loss event here.
    # For the sake of this example, we'll delete the source configuration file.
    try:
        os.remove(os.path.join(source_directory, "config.txt"))
        print("Source configuration file deleted.")
    except FileNotFoundError:
        pass
```



```python
        # Copy the configuration file to the backup directory
        shutil.copy(os.path.join(source_directory, "config.txt"), os.path.join(backup_directory, "config_backup.txt"))

        print("Backup completed successfully.")
    except Exception as e:
        print("Backup failed:", str(e))

# Function to recover data
def perform_recovery():
    try:
        # Copy the backed-up configuration file back to the source directory
        shutil.copy(os.path.join(backup_directory, "config_backup.txt"), os.path.join(source_directory, "config.txt"))

        print("Recovery completed successfully.")
    except Exception as e:
        print("Recovery failed:", str(e))

# Simulate a disaster recovery scenario
def simulate_disaster():
    print("Simulating a disaster...")
    # In a real-world scenario, you might encounter a data loss event here.
    # For the sake of this example, we'll delete the source configuration file.
    try:
        os.remove(os.path.join(source_directory, "config.txt"))
        print("Source configuration file deleted.")
    except FileNotFoundError:
        pass

# Main program
if __name__ == "__main__":
    # Perform an initial backup
    perform_backup()

    # Simulate a disaster by deleting the source configuration file
    simulate_disaster()

    # Perform data recovery
    perform_recovery()
```

OUTPUT



```
AMD64)] on win32
Type "help", "copyright", "credits" or
>
==== RESTART: C:/Users/ELCOT/AppData/Lc
Backup completed successfully.
Simulating a disaster...
Source configuration file deleted.
Recovery completed successfully.
```

FUTURE GOALS:

- Improve the automation of the disaster recovery process. This would help to reduce the time it takes to recover from a disaster and minimize downtime.
- Develop new algorithms to improve the performance and reliability of disaster recovery. This would help to ensure that data and applications are available and accessible even in the event of a major disaster.
- Make disaster recovery more affordable and accessible to businesses of all sizes. This would help to protect more businesses from the financial impact of disasters.

By achieving these goals, IBM Cloud VS can become the leading cloud-based disaster recovery solution for businesses of all sizes.

CONCLUSION:

Disaster recovery with IBM Cloud Virtual Servers (VS) is a reliable and efficient way to protect your data and applications from disasters. IBM Cloud VS uses a variety of features and services, as well as sophisticated algorithms, to ensure that your data and applications are available and accessible even in the event of a disaster.