Cloud-Based Disaster Recovery System

PHASE 3 : COMMUNICATION AND FUTURE EXPLORATION

ABSTRACT:

Disaster Recovery as a Service (DRaaS) is a lifeline for businesses, providing a safety net in the unpredictable world of data management. It is a thirdparty service that responds to disasters by protecting data and seamlessly aiding recovery operations. In layman's words, it's similar to having an emergency plan ready to go at any time, but with experts taking care of the details.

At its foundation, DRaaS is based on a subscription model, with organizations paying for the service as they use it. This means companies don't have to spend considerably in infrastructure up front, making it a more cost-effective option, particularly for smaller enterprises with limited resources. The beauty of DRaaS is its capacity to replicate and host both physical and virtual servers. So, Whether it's a cyberattack, a natural disaster, or any unforeseen incident that causes havoc, DRaaS ensures that operations can seamlessly transition to a secondary system.

But how does the magic happen? It is all about failover and failback. When the primary system fails, failover is the process of seamlessly moving operations to a backup system. It serves as a safety net to keep enterprises from entering a state of indefinite downtime. Once the storm has passed and the primary system is back up and running, failback takes over, seamlessly moving operations back to where they belong. The goal is to minimize disturbance and keep business running as usual, or as close to it as possible, even in the face of adversity.

The emergence of DRaaS is unsurprising given the growing awareness of data security in the business community. Companies recognize that their data is their lifeline, and any threats to it can have serious effects. They may be confident that their data is safe when they outsource disaster recovery planning to experts. According to Global Market Insights, the DRaaS market was worth USD 11.5 billion in 2022, with a projected 22% growth rate the following year. These figures highlight the growing importance of DRaaS in today's corporate world.

So, what is a disaster recovery plan (DRP), and why is it important? Consider it a crisis response plan. It defines how a business will respond to

unexpected occurrences, guaranteeing that they can recover quickly and efficiently. Along with business continuity plans (BCPs), disaster recovery plans (DRPs) form the backbone of a company's resilience strategy, assisting them in weathering disasters.

Effective DRPs require thorough planning and preparation. They include completing risk assessments, identifying essential assets, assigning roles and duties, and testing the plan on a regular basis to verify it is effective. In the event of a disaster, having a strong DRP in place can be the difference between a quick recovery and extended downtime, possibly saving firms millions in lost income and reputation harm.

However, DRaaS is more than just reacting to calamities; it is also about taking proactive steps to preserve your data. This is where backup as a service (BaaS) comes in handy. While DRaaS covers both data and infrastructure, BaaS is just about storing your data in secure, offsite places.

It's similar to having a safety deposit box for your most precious digital assets, ensuring that they remain safe at all times.

Choosing the correct backup location is critical in the DRaaS journey. Whether you choose a data center, the cloud, or a hybrid configuration, each has advantages and disadvantages. Data center backups provide physical security, but they can be expensive and time-consuming to recover. Cloud backups, on the other hand, are scalable and cost-effective, but they might create privacy and security concerns. Hybrid systems combine the benefits of cloud scalability with on-premises infrastructure security.

Finally, the purpose of DRaaS is to provide organizations piece of mind by ensuring that their data is safe and secure no matter what happens. By outsourcing disaster recovery planning to experts, they can concentrate on what they do best, knowing that their digital lifeline is secure.

The DRaaS industry is rapidly expanding, indicating that organizations understand its importance in today's data-driven environment.

SYSTEM REQUIREMENTS:

HARDWARE REQUIREMENTS

1. Processor: Dual-core processor or higher.
2. RAM-4 GB or higher.
3. Storage : oci Cloud storage.

4. Network: Stable internet connection with sufficient bandwidth for data replication and failover operations.

SOFTWARE REQUIREMENTS

1. Operating System: Linux (preferred) or Windows Server.
2. Virtualization Platform: Software like VMware, Hyper-V, or VirtualBox for creating virtual machines (if virtualization is used for hosting servers).
3. Cloud Services: Subscription to a reliable cloud service provider for hosting backup infrastructure and facilitating disaster recovery operations.

TOOLS AND VERSIONS:

1. Operating System:

   - Linux Distribution: Latest LTS version

2. Virtualization Platformx :

   - VMware vSphere: 7.0 or later

   - Microsoft Hyper-V: Windows Server 2019 or later

3. Cloud Service Providers:

   - OCI: Latest version

4. Networking Equipment:

   - Router/Firewall: Recommended version

5. Security Measures:

   - Antivirus/Anti-malware Software: Latest version

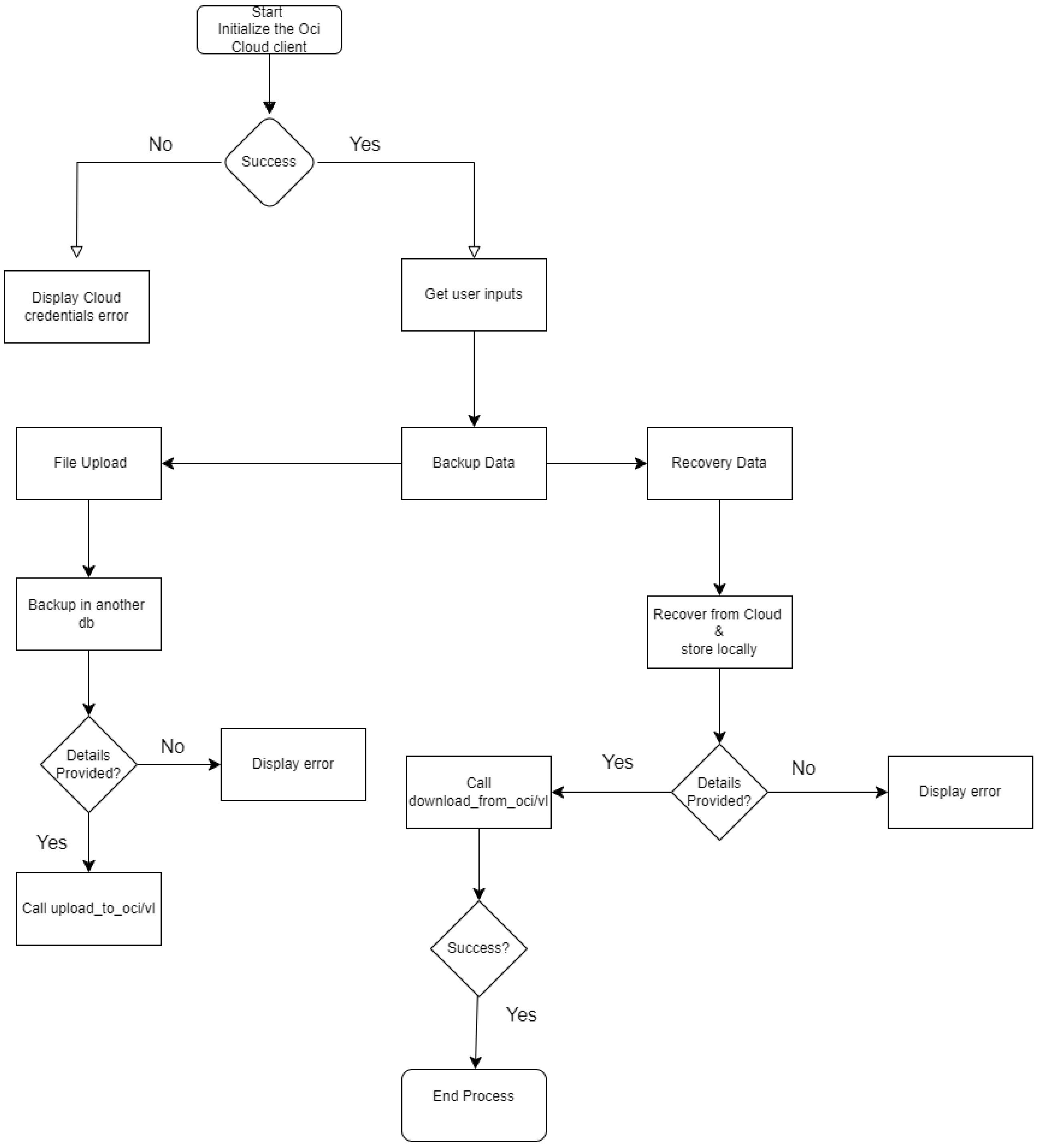
   - Encryption Tools: Latest version

6. Version Control:

- Git: 2.33 or later
- GitHub/GitLab/Bitbucket: Latest version

8. Development Frameworks and Languages:
   - Python
   - Stream-lit
   - OCI Cloud Database/ Storage

FLOWCHART:


Start
Initialize the Oci
Cloud client
Success
No
Yes
Display Cloud
credentials error
Get user inputs
File Upload
Backup Data
Recovery Data
Backup in another
db
Details
Provided?
No
Display error
Yes
Call upload_to_oci/vl
Recover from Cloud
&
store locally
Yes
Details
Provided?
No
Display error
Call
download_from_oci/vl
Success?
Yes
End Process

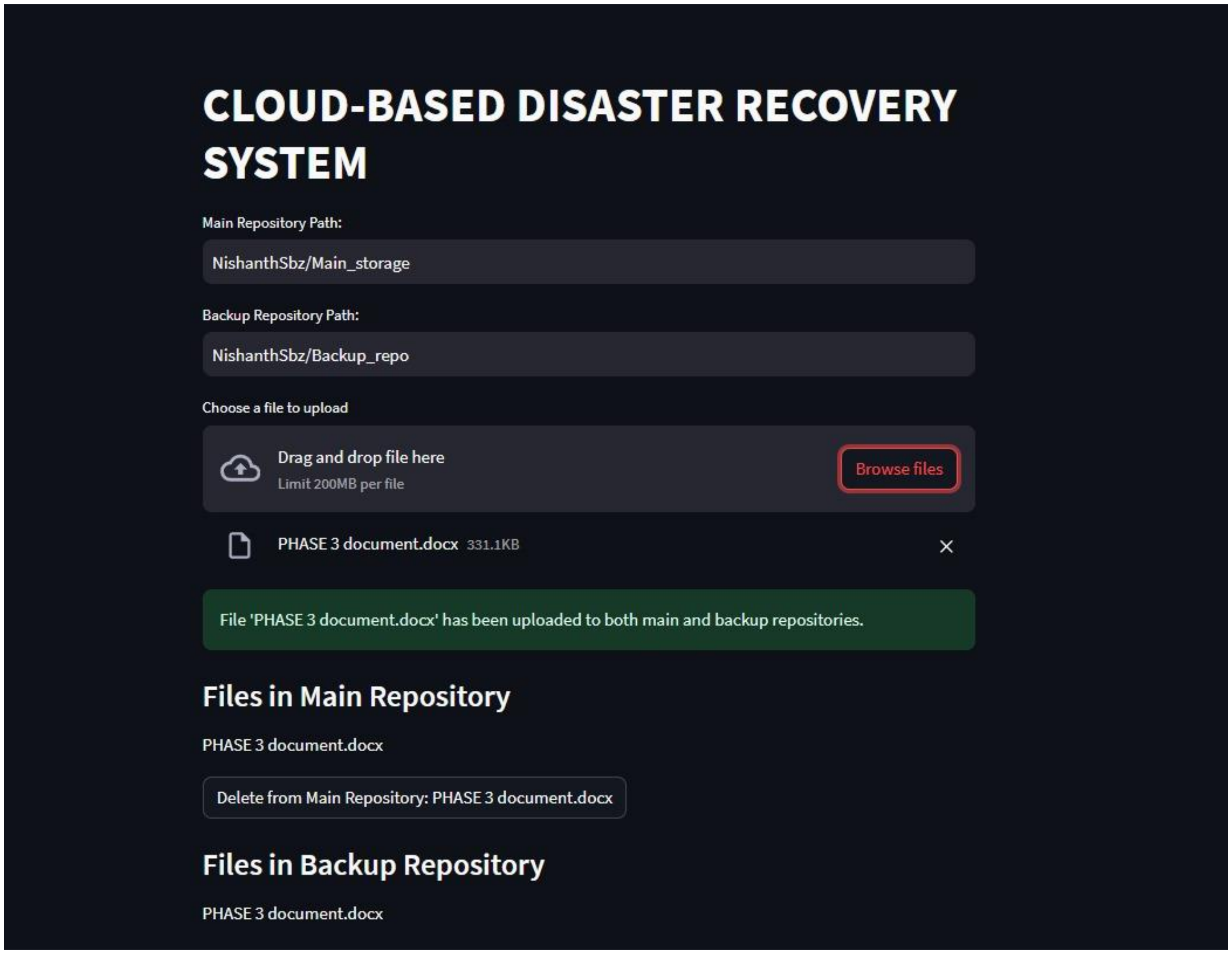
CODE IMPLEMENTATION(SAMPLE CODE):

```
import streamlit as st
import requests
import base64

# GitHub configuration
GITHUB_TOKEN = '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'  #
Replace with your GitHub PAT
MAIN_REPO = 'NishanthSbz/Main_storage'  # Replace with your GitHub username
and repository for main storage
BACKUP_REPO = 'NishanthSbz/Backup_repo'  # Replace with your GitHub
username and repository for backup storage

headers = {
    "Authorization": f"Bearer {GITHUB_TOKEN}",
    "Accept": "application/vnd.github.v3+json"
}

def upload_to_github(file, repo, path):
    try:
        url = f"https://api.github.com/repos/{repo}/contents/{path}/{file.name}"
        content = base64.b64encode(file.getvalue()).decode()
        data = {
            "message": f"Add {file.name}",
            "content": content
        }
        response = requests.put(url, headers=headers, json=data)
        response.raise_for_status()
        return response.json()['content']['path']
    except Exception as e:
        st.error(f"Failed to upload to GitHub: {e}")
        return None

def list_files_in_github(repo, path):
    try:
        url = f"https://api.github.com/repos/{repo}/contents/{path}"
        response = requests.get(url, headers=headers)
        response.raise_for_status()
        files = response.json()
        return [file['name'] for file in files if file['type'] == 'file']
    except Exception as e:
        st.error(f"Failed to list files in GitHub: {e}")
        return []

def delete_from_github(repo, path, file_name):
    try:
        url = f"https://api.github.com/repos/{repo}/contents/{path}/{file_name}"
        get_file_response = requests.get(url, headers=headers)
        get_file_response.raise_for_status()
        sha = get_file_response.json()['sha']
        data = {
            "message": f"Delete {file_name}",
            "sha": sha
        }
        delete_response = requests.delete(url, headers=headers, json=data)
        delete_response.raise_for_status()
    except Exception as e:
        st.error(f"Failed to delete file from GitHub: {e}")

# Streamlit UI
st.title("CLOUD-BASED DISASTER RECOVERY SYSTEM")

# Input GitHub repository paths for main and backup storage
main_repo_path = st.text_input("Main Repository Path:", MAIN_REPO)
backup_repo_path = st.text_input("Backup Repository Path:", BACKUP_REPO)

if main_repo_path and backup_repo_path:
    # File uploader
    uploaded_file = st.file_uploader("Choose a file to upload")

    if uploaded_file is not None:
        # Save file to main and backup GitHub repositories
        main_path = upload_to_github(uploaded_file, main_repo_path, 'main')
        backup_path = upload_to_github(uploaded_file, backup_repo_path, 'backup')

        if main_path and backup_path:
            st.success(f"File '{uploaded_file.name}' has been uploaded to both main and
backup repositories.")

    # Display files in main repository
    st.subheader("Files in Main Repository")
    main_files = list_files_in_github(main_repo_path, 'main')
    for file_name in main_files:
        st.write(file_name)
        if st.button(f"Delete from Main Repository: {file_name}",
key=f"delete_main_{file_name}"):
            delete_from_github(main_repo_path, 'main', file_name)
            st.warning(f"File '{file_name}' deleted from main repository but remains in
backup repository.")

    # Display files in backup repository
    st.subheader("Files in Backup Repository")
    backup_files = list_files_in_github(backup_repo_path, 'backup')
    for file_name in backup_files:
        st.write(file_name)
else:
    st.error("Please provide both main and backup repository paths.")
```

PROJECT HURDLES:

During the execution of our project, we faced several challenges. Integrating OCI cloud storage with our Streamlit application proved especially tough due to credential management and guaranteeing secure access. Debugging issues in file uploads and downloads necessitated substantial debugging. Furthermore, coordinating work across team members with different schedules and technical competence hindered our progress. Despite these challenges, we were able to strengthen our problemsolving skills and awareness of cloud services.

OUTPUT:



CLOUD-BASED DISASTER RECOVERY SYSTEM

Main Repository Path:

NishanthSbz/Main_storage

Backup Repository Path:

NishanthSbz/Backup_repo

Choose a file to upload

Drag and drop file here
Limit 200MB per file

Browse files

PHASE 3 document.docx 331.1KB

File 'PHASE 3 document.docx' has been uploaded to both main and backup repositories.

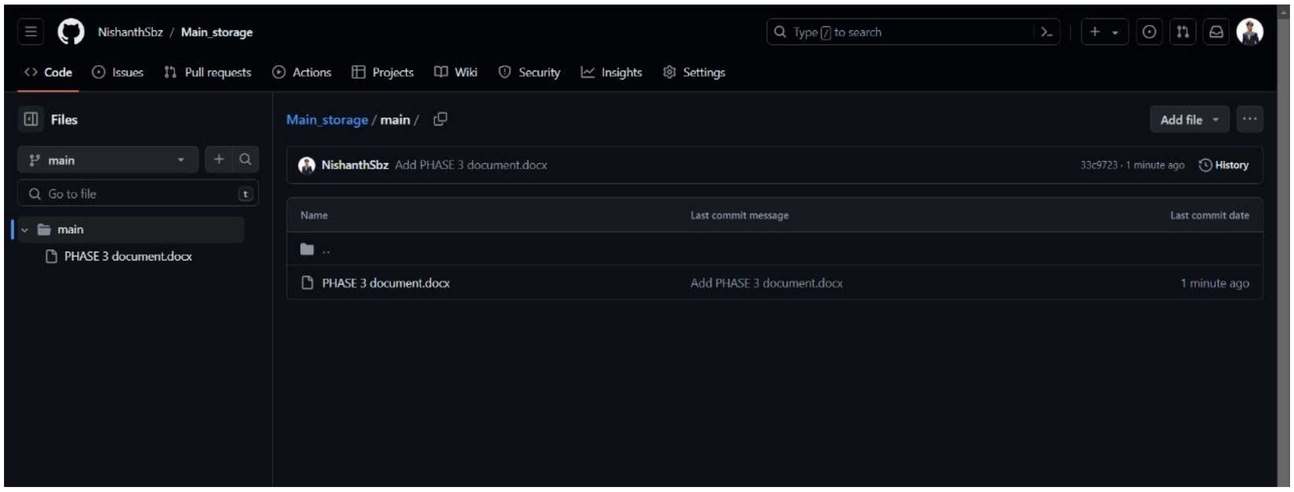
Files in Main Repository

PHASE 3 document.docx

Delete from Main Repository: PHASE 3 document.docx

Files in Backup Repository

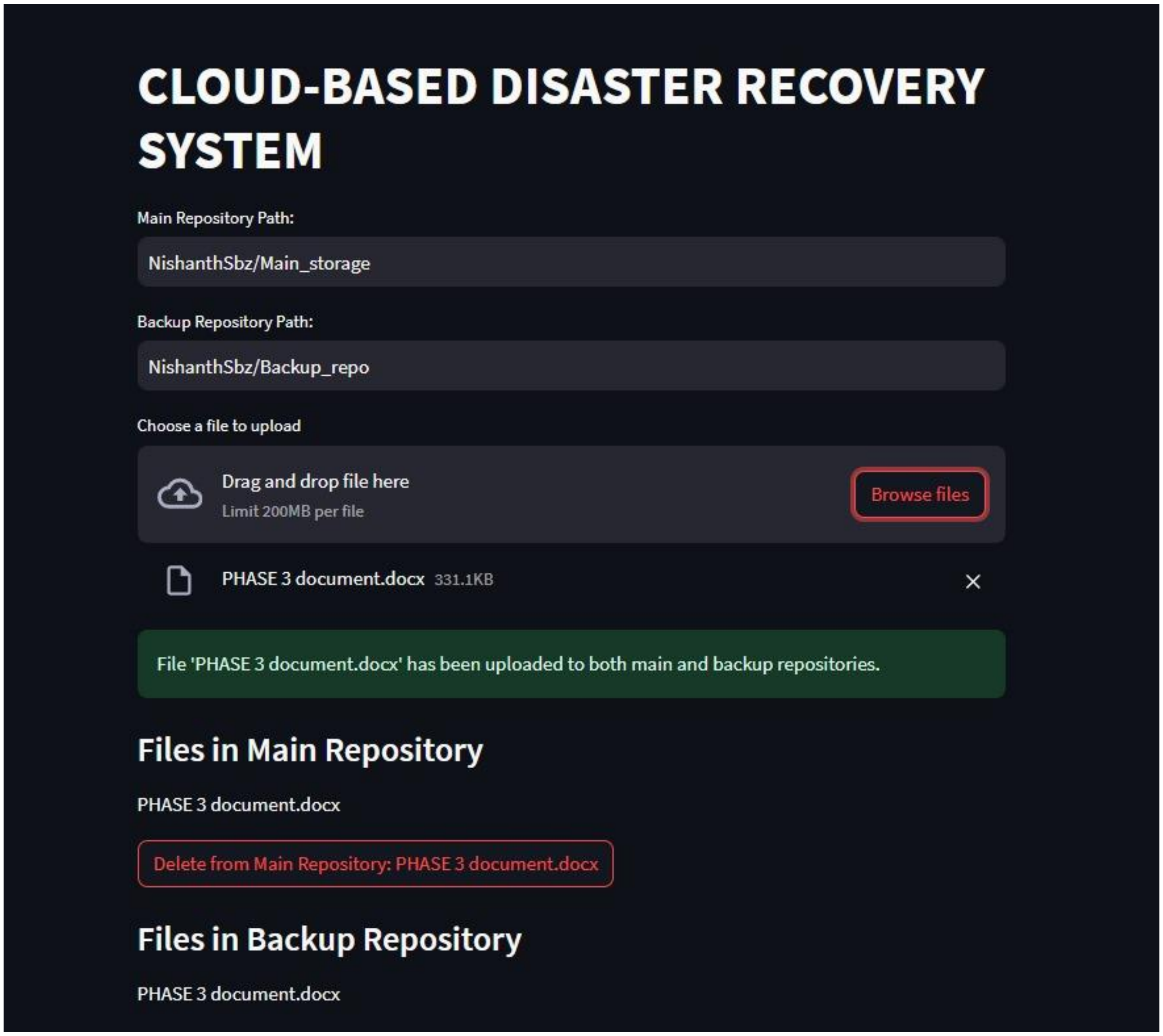
PHASE 3 document.docx



Main_storage / main /

NishanthSbz Add PHASE 3 document.docx 33c9723 · 1 minute ago History

Name	Last commit message	Last commit date
..		
PHASE 3 document.docx	Add PHASE 3 document.docx	1 minute ago


CLOUD-BASED DISASTER RECOVERY SYSTEM

Main Repository Path:

NishanthSbz/Main_storage

Backup Repository Path:

NishanthSbz/Backup_repo

Choose a file to upload

PHASE 3 document.docx 331.1KB

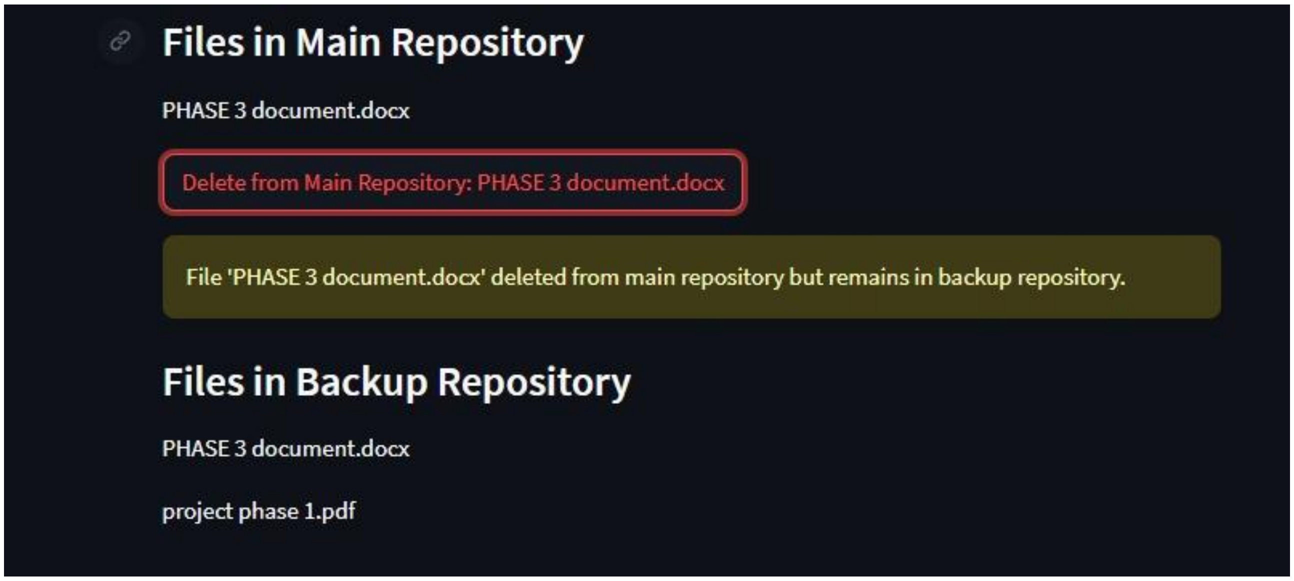
File 'PHASE 3 document.docx' has been uploaded to both main and backup repositories.

Files in Main Repository

PHASE 3 document.docx

Files in Backup Repository

PHASE 3 document.docx


Files in Main Repository

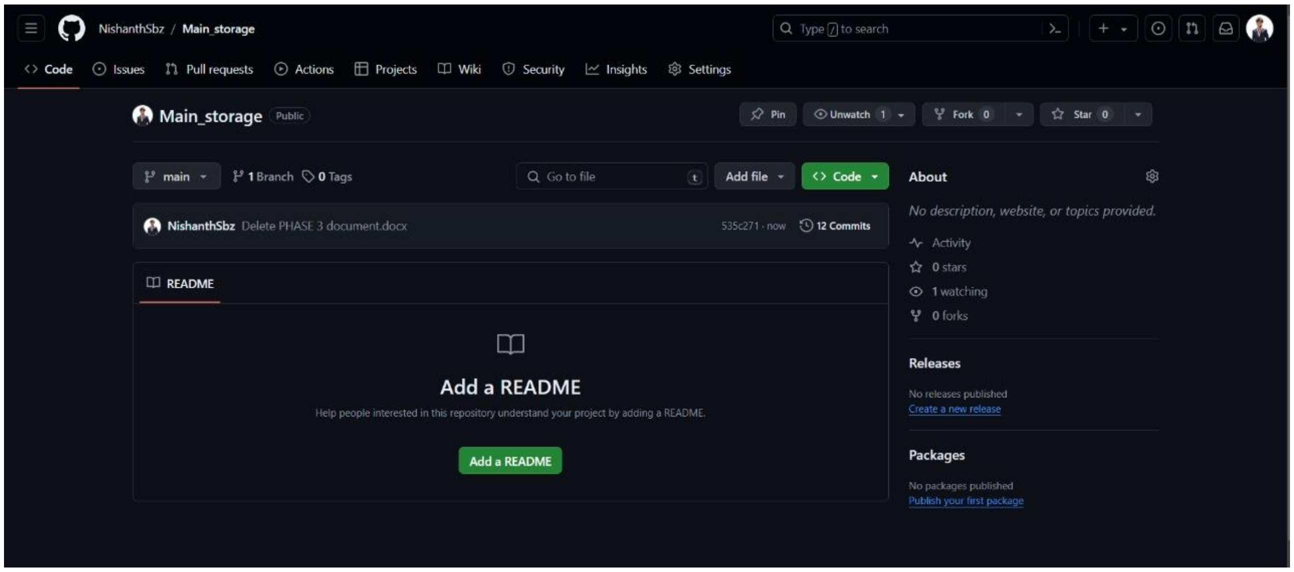
PHASE 3 document.docx

File 'PHASE 3 document.docx' deleted from main repository but remains in backup repository.

Files in Backup Repository

PHASE 3 document.docx

project phase 1.pdf



NishanthSbz Delete PHASE 3 document.docx 535c271 · now 12 Commits

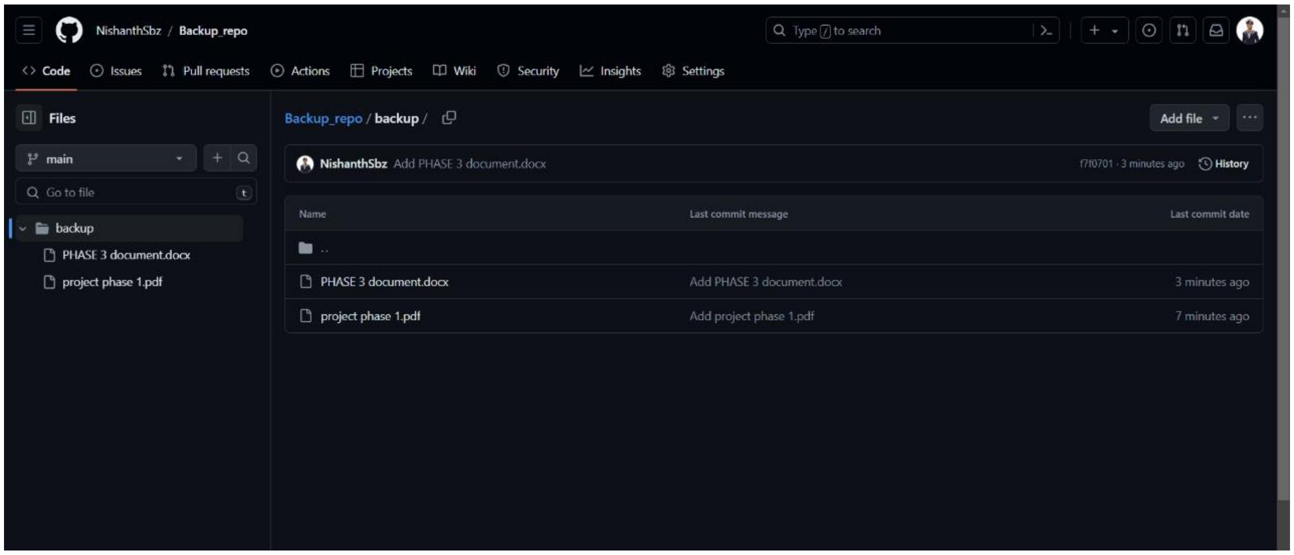
Help people interested in this repository understand your project by adding a README.

About

No description, website, or topics provided.

No releases published

No packages published



NishanthSbz Add PHASE 3 document.docx f7f0701 · 3 minutes ago

Name	Last commit message	Last commit date
..		
PHASE 3 document.docx	Add PHASE 3 document.docx	3 minutes ago
project phase 1.pdf	Add project phase 1.pdf	7 minutes ago

CONCLUSION AND FUTURE SCOPE:

To summarize, Disaster Recovery as a Service (DRaaS) provides a critical safety net for businesses, ensuring that data is safeguarded and recovery processes run smoothly in the case of disaster. DRaaS runs on a subscription basis, making it an affordable alternative, particularly for smaller businesses. Its capacity to replicate and host both physical and virtual servers means that enterprises may continue to operate even amid bad occurrences. The essential principles of failover and failback allow operations to seamlessly move between primary and secondary systems, reducing downtime and interruption.

DRaaS deployment involves a number of components and considerations. A strong disaster recovery plan (DRP) is required, defining how a company will respond to unanticipated catastrophes. Effective DRPs necessitate meticulous preparation, including risk assessments, asset identification, role allocations, and regular testing. Backup as a Service (BaaS) complements DRaaS by focusing on safely storing data offsite.

Our project entailed creating a cloud-based disaster recovery system with AWS S3 and Streamlit. We specified the hardware and software requirements, including the required tools and versions. The study emphasized the significance of having the necessary infrastructure and tools in place for efficient data replication and recovery. One of the most difficult issues we encountered was integrating AWS S3 with our Streamlit application. Managing credentials securely and ensuring smooth file uploads and downloads necessitated extensive testing and problem-solving. Furthermore, coordinating work among team members with different schedules and technical knowledge presented difficulties, but ultimately increased our teamwork and technical skills.

Looking ahead, the scope of this endeavor is broad. As the DRaaS market expands, driven by growing awareness of data security, there are various opportunities for continued development and enhancement:

1. Enhanced Security Measures: Using advanced security protocols like multi-factor authentication and encryption at rest and in transit can help secure data during replication and recovery.

2. AI and machine learning techniques can help improve disaster recovery prediction analytics. These technologies can assist identify possible risks and optimize recovery tactics, decreasing downtime and increasing efficiency.

3. Automation and Orchestration: Creating automated workflows for failover and failback operations can reduce the need for human intervention and the risk of errors. Orchestration solutions help speed up the recovery process by guaranteeing a smooth transition between primary and secondary systems.

4. Scalability and Flexibility: Extending the system to handle multi-cloud environments can improve flexibility and redundancy. This would enable enterprises to use the qualities of many cloud providers, resulting in increased availability and reliability.
5. User Experience Improvements: By improving the disaster recovery application's user interface and experience, it can become more intuitive and accessible. Real-time monitoring, rich analytics, and customisable dashboards

can help customers gain more control and visibility over their recovery procedures.

6. Compliance and Regulatory Support: It is critical to ensure that the DRaaS solution meets all relevant regulatory requirements and industry standards. Implementing tools that enable compliance reporting and audit trails can assist firms in meeting their legal and regulatory duties.

To summarize, creating a cloud-based disaster recovery system is an important step toward ensuring business continuity in the case of disasters. Businesses may protect their digital assets and keep operations running smoothly by embracing new technology and constantly improving the system's capabilities. The future of DRaaS seems exciting, with multiple prospects for innovation and development that will ultimately provide organizations with more peace of mind and resilience.