DISASTER RECOVERY PLAN WITH IBM CLOUD SERVICES

Introduction :

- Brief overview of the project
- Importance of Disaster Recovery Planning

1. Disaster Recovery Strategy :

- Definition of RTO, RPO, and Priority
- Components: Virtual Servers, DB2, Object Storage, Watson Assistant

2. Setting Up Regular Backups :

- Virtual Machines
- IBM DB2
- IBM Object Storage
- IBM Watson Assistant

3. Testing and Drills :

- Importance of Testing
- Conducting Disaster Recovery Drills

4. Documentation and Communication :

- Documenting Procedures
- Communication Channels

5. Monitoring and Alerting :

- Tools and Techniques
- Proactive Measures

6. Review and Update :

- Continuous Improvement
- Adapting to Changes

7. Security Measures :

- Encryption and Access Controls
- Compliance Considerations

1. Disaster Recovery Strategy :

- RTO (Recovery Time Objective)
- Definition
- Example: Virtual Machines vs. DB2
- RPO (Recovery Point Objective)
- Definition

Example: Watson Assistant vs. Object Storage

- Priority of Components
- Categorization
- Critical vs. Non-Critical

2. Setting Up Regular Backups

- Virtual Machines
- Using IBM Cloud Tools
- Custom Scripts for Automation

IBM DB2

- Backup and Recovery Strategies
- Scheduled Backups

IBM Object Storage

- Durability and Redundancy
- Replication for Critical Data

IBM Watson Assistant

- Exporting Configurations
- Automated Backup Scripts

3. Testing and Drills

- Importance of Testing
- Ensuring Effectiveness
- Identifying Gaps
- Conducting Disaster Recovery Drills
- Simulating Disaster Scenarios
- Evaluating Recovery Procedures

4. Documentation and Communication

- Documenting Procedures
- Detailed Step-by-Step Instructions
- Contact Information
- Communication Channels
- Notifying Stakeholders
- Establishing Clear Lines of Communication

5. Monitoring and Alerting

- Monitoring Tools
- Ensuring Service Health
- Tracking Backup Status
- Proactive Measures
- Setting up Alerts
- Response Plans for Failures

6. Review and Update

- Continuous Improvement
- Periodic Reviews
- Adapting to Changing Requirements
- Adapting to Changes
- Updates in Technology
- Business Process Changes

7. Security Measures

- Encryption
- Data at Rest and in Transit
- Compliance Considerations
- Access Controls
- Preventing Unauthorized Access
- Role-Based Access Policies

8.Conclusion

- Recap of Key Points
- Importance of a Robust Disaster Recovery Plan

```
from flask import Flask, render_template, request

app = Flask(__name__)

disaster_recovery_plan = {
    "RTO": None,
    "RPO": None,
    "priority": None,
    "backup_tool": None
}

virtual_servers = []

db2_servers = []

object_storage = []

watson_assistant = []

backup_schedule = []
```

```
@app.route('/')
def index():
 return render_template('index.html', plan=disaster_recovery_plan,
                        servers=virtual_servers, db2=db2_servers,
                        storage=object_storage, assistant=watson_assistant,
                        backups=backup_schedule)
@app.route('/update_plan', methods=['POST'])
def update_plan():
    disaster_recovery_plan['RTO'] = request.form['RTO']
    disaster_recovery_plan['RPO'] = request.form['RPO']
    disaster_recovery_plan['priority'] = request.form['priority']
    disaster_recovery_plan['backup_tool'] = request.form['backup_tool']
    return render_template('index.html', plan=disaster_recovery_plan,
servers=virtual_servers, db2=db2_servers,
                        storage=object_storage, assistant=watson_assistant,
                        backups=backup_schedule)
@app.route('/add_server', methods=['POST'])
def add_server():
    server_name = request.form['server_name']
    virtual_servers.append(server_name)
    return render_template('index.html', plan=disaster_recovery_plan,
                        servers=virtual_servers, db2=db2_servers,
                        storage=object_storage, assistant=watson_assistant,
                        backups=backup_schedule)
```

```
@app.route('/add_db2', methods=['POST'])
def add_db2():
    db2_name = request.form['db2_name']
    db2_servers.append(db2_name)
    return render_template('index.html', plan=disaster_recovery_plan,
                          servers=virtual_servers, db2=db2_servers,
                          storage=object_storage, assistant=watson_assistant,
                          backups=backup_schedule)
@app.route('/add_storage', methods=['POST'])
def add_storage():
    storage_name = request.form['storage_name']
    object_storage.append(storage_name)
    return render_template('index.html', plan=disaster_recovery_plan,
                          servers=virtual_servers, db2=db2_servers,
 storage=object_storage, assistant=watson_assistant,
                           backups=backup_schedule)
@app.route('/add_assistant', methods=['POST'])
def add_assistant():
    assistant_name = request.form['assistant_name']
    watson_assistant.append(assistant_name)
    return render_template('index.html', plan=disaster_recovery_plan,
                           servers=virtual_servers, db2=db2_servers,
                           storage=object_storage, assistant=watson_assistant,
                           backups=backup_schedule)
```

```
@app.route('/view_plan')
def view_plan():
    return render_template('view_plan.html', plan=disaster_recovery_plan,
                           servers=virtual_servers, db2=db2_servers,
                           storage=object_storage, assistant=watson_assistant,
                           backups=backup_schedule)
@app.route('/update_priority', methods=['POST'])
def update_priority():
    server_name = request.form['server_name']
```

```
 priority = request.form['priority']
for server in virtual_servers:
    if server == server_name:
        disaster_recovery_plan['priority'] = priority
        break
 return redirect('/view_plan’)
@app.route('/add_backup', methods=['POST'])
def add_backup():
 backup_name = request.form['backup_name']
    backups.append(backup_name)
    return redirect('/view_plan')
@app.errorhandler(404)
def page_not_found(e):
    return "404 Page Not Found", 404
def save_data():
    with open('data.txt', 'w') as file:
        file.write(f"{disaster_recovery_plan}\n")
        file.write(f"{virtual_servers}\n")
        file.write(f"{db2_servers}\n")
        file.write(f"{object_storage}\n")
        file.write(f"{watson_assistant}\n")
        file.write(f"{backup_schedule}\n")
```

```
def load_data():
    try:
        with open('data.txt', 'r') as file:
            data = file.readlines()
            global disaster_recovery_plan, virtual_servers, db2_servers, object_storage,
watson_assistant, backup_schedule
            disaster_recovery_plan = eval(data[0])
            virtual_servers = eval(data[1])
            db2_servers = eval(data[2])
            object_storage = eval(data[3])
            watson_assistant = eval(data[4])
            backup_schedule = eval(data[5])
    except FileNotFoundError:
        pass
```

```
def reset_data():
    global disaster_recovery_plan, virtual_servers, db2_servers, object_storage,
watson_assistant, backup_schedule
    disaster_recovery_plan = {
        "RTO": None,
        "RPO": None,
        "priority": None,
        "backup_tool": None
    }
    virtual_servers = []
    db2_servers = []
    object_storage = []
    watson_assistant = []
    backup_schedule = []
```

```
# ... Previous code ...
@app.route('/reset_data')
def reset_saved_data():
    reset_data()
    return redirect('/view_plan')


@app.route('/reset_success')
def reset_success():
    return "Data reset successfully. <a href='/'>Go back</a>."
if __name__ == '__main__':
    app.run(debug=True)
```

Conclusion

- Recap of Key Points
- Importance of a Robust Disaster Recovery Plan