Subject: Sending bcp outline ideas to client id
Date: Thursday, 6 February 2025 at 8:17:22 PM India Standard Time
From: Koppisetti, Venkata
To: venkata.koppisetti@bupa.com

Disaster Recovery (DR) strategies for **Terraform Enterprise (TFE)** are critical to ensure business continuity in case of infrastructure failures, data loss, or service disruptions. Below is a detailed guide on DR strategies for Terraform Enterprise:

Disaster Recovery Strategies for Terraform Enterprise

1. Key Components to Protect
- **State Files**: The most critical component, as they store the current state of your infrastructure.
- **Workspaces**: Configuration and metadata for Terraform workspaces.
- **Policies and Sentinel Policies**: Governance rules and policies.
- **Run Data**: Historical data about Terraform runs.
- **VCS Connections**: Version Control System (e.g., GitHub, GitLab) integrations.
- **Secrets and Variables**: Sensitive data stored in TFE.

---

2. Backup Strategies

A. State File Backups
1. **Remote State Storage**:
  - Use TFE's built-in remote state storage with redundancy (e.g., S3, Azure Blob, GCS).
  - Enable versioning on the storage backend to recover previous state file versions.

2. **Automated State Backups**:
  - Use Terraform's `terraform state pull` to manually back up state files:
    ```bash
    terraform state pull > state_backup.tfstate
    ```
  - Automate this process using scripts or CI/CD pipelines.

3. **TFE API for State Export**:
  - Use the TFE API to export state files:
    ```bash
    curl -H "Authorization: Bearer $TOKEN" \
    https://app.terraform.io/api/v2/workspaces/$WORKSPACE_ID/current-state-version \
    -o state_backup.tfstate
    ```

B. Workspace and Run Data Backups
1. **TFE API for Workspace Data**:
  - Export workspace configurations and run data using the TFE API.

- Example:
  ```bash
  curl -H "Authorization: Bearer $TOKEN" \
  https://app.terraform.io/api/v2/workspaces/$WORKSPACE_ID \
  -o workspace_backup.json
  ```

2. **Database Backups**:
   - For self-hosted TFE, back up the PostgreSQL database regularly:
     ```bash
     pg_dump -U tfe_user -h tfe_db_host -d tfe_db -f tfe_backup.sql
     ```

C. Sentinel Policies and Configurations
1. **Version Control**:
   - Store Sentinel policies and configurations in a version control system (e.g., GitHub, GitLab).
   - Use `git` to clone and back up repositories:
     ```bash
     git clone https://github.com/your-org/sentinel-policies.git
     ```

2. **TFE API for Policy Sets**:
   - Export policy sets using the TFE API:
     ```bash
     curl -H "Authorization: Bearer $TOKEN" \
     https://app.terraform.io/api/v2/organizations/$ORG/policy-sets \
     -o policy_sets_backup.json
     ```

D. Secrets and Variables
1. **External Secrets Management**:
   - Use tools like HashiCorp Vault, AWS Secrets Manager, or Azure Key Vault to store sensitive data.
   - Reference secrets in TFE using environment variables or workspace variables.

2. **Export Variables**:
   - Use the TFE API to export workspace variables:
     ```bash
     curl -H "Authorization: Bearer $TOKEN" \
     https://app.terraform.io/api/v2/workspaces/$WORKSPACE_ID/vars \
     -o variables_backup.json
     ```

---

3. Recovery Strategies

A. State File Recovery
1. **Restore from Remote Storage**:
   - Recover state files from versioned storage (e.g., S3, Azure Blob).

2. **Manual State Upload**:
   - Use `terraform state push` to restore a state file:
     ```bash
     terraform state push state_backup.tfstate
     ```

3. **TFE API for State Import**:
   - Use the TFE API to import a state file:
     ```bash
     curl -H "Authorization: Bearer $TOKEN" \
     -H "Content-Type: application/json" \
     -d @state_backup.tfstate \
     https://app.terraform.io/api/v2/workspaces/$WORKSPACE_ID/state-versions
     ```

B. Workspace and Run Data Recovery
1. **Restore from Database Backup**:
   - For self-hosted TFE, restore the PostgreSQL database:
     ```bash
     psql -U tfe_user -h tfe_db_host -d tfe_db -f tfe_backup.sql
     ```

2. **Recreate Workspaces**:
   - Use the TFE API or Terraform provider to recreate workspaces from backups.

C. Sentinel Policies and Configurations
1. **Reapply Policies**:
   - Reapply Sentinel policies from version control or backups.

2. **Reimport Policy Sets**:
   - Use the TFE API to reimport policy sets:
     ```bash
     curl -H "Authorization: Bearer $TOKEN" \
     -H "Content-Type: application/json" \
     -d @policy_sets_backup.json \
     https://app.terraform.io/api/v2/organizations/$ORG/policy-sets
     ```

D. Secrets and Variables
1. **Reimport Variables**:
   - Use the TFE API to reimport workspace variables:
     ```bash
     curl -H "Authorization: Bearer $TOKEN" \
     -H "Content-Type: application/json" \
     -d @variables_backup.json \
     https://app.terraform.io/api/v2/workspaces/$WORKSPACE_ID/vars
     ```

---

4. High Availability (HA) for Self-Hosted TFE

1. **Active/Active Clustering**:
  - Deploy TFE in an active/active configuration for redundancy.
  - Use a load balancer to distribute traffic across multiple TFE instances.

2. **Database Replication**:
  - Set up PostgreSQL replication for high availability.

3. **Object Storage Redundancy**:
  - Use multi-region object storage for state files and backups.

---

5. Testing and Validation
1. **Regular DR Drills**:
  - Simulate failures and test recovery processes quarterly.
  - Validate backup integrity and restoration workflows.

2. **Automated Testing**:
  - Use CI/CD pipelines to automate backup and recovery testing.

---

6. Monitoring and Alerts
1. **TFE Health Checks**:
  - Monitor TFE health using built-in health checks or external tools (e.g., Prometheus, Datadog).

2. **Backup Alerts**:
  - Set up alerts for backup failures or anomalies.

---

7. Documentation and Runbooks
1. **DR Runbook**:
  - Document step-by-step recovery procedures for each component.
  - Include contact information for key personnel.

2. **Training**:
  - Train staff on DR procedures and tools.

---

8. Third-Party Tools
Tool	Purpose
HashiCorp Vault	Secrets management and encryption.
AWS S3/Azure Blob/GCS	Redundant state file storage.
pgBackRest	PostgreSQL backup and recovery.
Terraform Cloud	Managed TFE with built-in redundancy.

Summary
- **Backup**: Regularly back up state files, workspaces, policies, and secrets.
- **Recovery**: Use TFE APIs, Terraform commands, and database backups for restoration.
- **HA**: Deploy TFE in an active/active configuration with redundant storage.
- **Testing**: Conduct regular DR drills and automate testing.

By implementing these strategies, you can ensure minimal downtime and data loss in the event of a disaster affecting Terraform Enterprise.