



TFE Technical Enablement

TFE Day to Day



Table of Contents



- TFE Monitoring
 - I/O
 - RAM
 - CPU
 - Disk
- Logging
 - Application Logs
 - Audit Logs
 - Log Formating
- Upgrades
- Backups and Restores
- Supporting TFE



Terraform Enterprise Architecture

Monitoring



TFE Monitoring



- Health Check
 - Terraform Enterprise provides a `/_health_check` endpoint on the instance. If Terraform Enterprise is up, the health check will return a 200 OK.
- Metrics & Telemetry
 - I/O
 - RAM
 - CPU
 - Disk

Internal Monitoring



Terraform Enterprise includes internal monitoring of critical application metrics using a statsd collector.

To disable metrics collection by Terraform Enterprise:

- Access the installer dashboard on port 8800 of your instance.
- Locate Advanced Configuration on the configuration page under Terraform Build Worker image.
- Uncheck "Enable metrics collection".
- Restart the application from the dashboard if it does not restart automatically.



Terraform Enterprise Architecture

Logging



Terraform Enterprise Logs



There are two types of logs

- **Application logs:** emit information about the services that comprise Terraform Enterprise
- **Audit logs:** emit information whenever any resource managed by Terraform Enterprise is changed

Forwarding TFE Logs



- Terraform Enterprise runs in a set of Docker containers
- TFE provides built in log forwarding using Fluent Bit
- Common destinations include Splunk, AWS CloudWatch, GCP Cloud logging etc.




TFE Log Forwarding Requirements



- TFE host has `systemd-journald` enabled (common default)
`systemctl status systemd-journald`
- Docker version with `journald` logging capabilities (default with install script)
`docker info --format '{{.Plugins.Log}}'`
- Fluent Bit configuration and connectivity to configured end point

TFE Log Forwarding Configuration



 **Terraform**
ENTERPRISE

Log Forwarding

Forward Terraform Enterprise logs to one or more external destinations.

☒ Enable Log Forwarding

Log Forwarding Configuration (Required)

```
# Example Fluent Bit configuration that matches all logs and does not
# forward them anywhere.
[OUTPUT]
  Name null
  Match *
```

Valid Fluent Bit **OUTPUT** configuration.

- Log forwarding can be configured from the **Admin Dashboard** or using the `replicatedctl cli`
- TFE must be restarted for changes to take effect
- Recommendation: Test Fluent Bit config using the `fluent/fluent-bit` image, only `OUTPUT` section of config is needed by TFE

Audit Logs



The audit logs are emitted along with other logs by the `ptfe_atlas` container. To distinguish audit log entries from other log entries, the log entry is prefixed with `[Audit Log]`. For example:

```
2020-12-15 12:20:37 [INFO] [ea6217bf-0495-4cbd-a477-304223db854a]  
[Audit Log] {"resource":"user","action":"create","resource_id":"admin",  
"organization":null,"actor":"AdminFirstUserAccountController",  
"timestamp":"2020-12-15T12:20:37Z","actor_ip":"207.59.136.92"}
```

Audit Logs – Log Contents



The audit log will be updated when any resource managed by Terraform Enterprise is changed. Read requests will be logged for resources deemed sensitive. These include:

- Authentication Tokens
- Configuration Versions
- Policy Versions
- OAuth Tokens
- SSH Keys
- State Versions
- Users
- Variables

Here's a Sample Log



```
{  
  "resource": "workspace",  
  "action": "update",  
  "resource_id": "ws-RCj9uT1aYH7F47LH",  
  "organization": "myorg",  
  "actor": "admin",  
  "timestamp": "2021-12-15T12:24:25Z",  
  "actor_ip": "207.59.136.92"  
}
```

What's in a Log



- The actor
 - Users (including IP address)
 - Version Control System users (identified in webhooks)
 - Service accounts
 - Terraform Enterprise

What's in a Log



- The action
 - Reading sensitive resources
 - Creation of new resources
 - Updating existing resources
 - Deletion of existing resources
 - Additional actions as defined in `/actions/*` namespaces
 - Webhook API calls

What's in a Log



- The target of the action (any resource exposed by the V2 API)
- The time that the action occurred
- Where the action was taken (web/API request, background job, etc.)



Terraform Enterprise Architecture

Upgrades



TFE Upgrading



Since Terraform Enterprise supports both online and offline installations there are multiple ways to upgrade.

- Online
 - From the installer dashboard (https://<TFE_HOSTNAME>:8800/dashboard), click the "Check Now" button; the new version should be recognized.
 - Click "View Update".
 - Review the release notes and then click "Install Update".

When an upgrade is ready to start the new code, the system waits for all Terraform runs to finish before continuing.

TFE Offline Upgrade



- Determine the update path where the installer will look for new .airgap packages. You can do this from the console settings of your instance (https://<TFE_HOSTNAME>:8800/console/settings) in the field Update Path.
- Download the new .airgap package onto the instance and put it into the Update Path location.
- From the installer dashboard (https://<TFE_HOSTNAME>:8800/dashboard) click the "Check Now" button; the new version should be recognized.
- Click "View Update".
- Review the release notes and then click "Install Update".



Terraform Enterprise Architecture

Backup and Restore



TFE Backups and Restores



Terraform Enterprise provides an API to backup and restore all of its application data.

The backup and restore API is separate from the Terraform Enterprise application-level APIs. As such, a separate authorization token is required to use the backup and restore API.

Using the backup and restore API is the only supported way to migrate between operational modes (mounted disk, external services).

About Backups and Restores



- The backup and restore API backs up all of the data stored in a Terraform Enterprise installation, including both the blob storage and the PostgreSQL database.
- This backup can then be restored to a new installation of Terraform Enterprise.
- It does not back up the installation configuration.

Things to Know

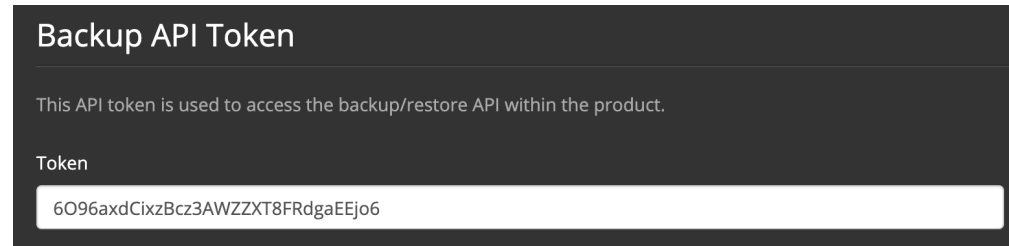


- A backup taken from one version of TFE cannot be restored to an installation running a different version
- The Terraform Enterprise installation that will be restored to must be a new installation with no existing application data
- Once a restore is completed, the application will need to be restarted before it can use the restored data

TFE - Backup and Restore Authentication



The backup and restore API uses a separate authorization token which can be found on the settings dashboard (https://<TFE_HOSTNAME>:8800/settings) near the bottom of the page:



The backup and restore API is separate from the Terraform Enterprise application-level APIs and cannot be accessed with Terraform Enterprise user, team, or organization API tokens.

To use this authorization token with the backup and restore API, pass the `Authorization: Bearer <TOKEN>` header in your API requests.

Security and Encryption in Backups



- Terraform Enterprise uses HashiCorp Vault to encrypt and decrypt its data.
- The Vault encryption keys that are used to encrypt and decrypt this data are not preserved during a backup or restore.
 - Instead, during a backup, the data is decrypted by Vault and then re-encrypted using a password provided by you, resulting in an encrypted backup blob.
- During a restore, the same password that you provided during the backup must be used to decrypt the data before it is re-encrypted with the new Terraform Enterprise installation's Vault encryption keys.

Security and Encryption in Backups



The backup and restore API expect this password to be provided as a JSON object with a password property within the request payload. The value for the password property can be any valid string.

Sample Payload:

```
{  
  "password": "happily-genuinely-heroic-sponge"  
}
```

Creating a Backup



To initiate a backup, make a POST request to the backup endpoint on a running Terraform Enterprise installation.

```
curl \
  --header "Authorization: Bearer $TOKEN" \
  --request POST \
  --data @payload.json \
  --output backup.blob \
  https://<TFE_HOSTNAME>/_backup/api/v1/backup
```

A successful backup must return 200. If 200 is not returned and the call silently closes, the backup blob may be incomplete, resulting in data loss.

Things to Note



- Remember to specify an output file for the encrypted backup blob.
- Be prepared to download and store many gigabytes of data to the filesystem of whichever machine the request is sent from.
- For best performance and to avoid disconnections, we recommend sending this request from a server colocated with the Terraform Enterprise installation rather than from a workstation.
- Treat this encrypted backup blob as sensitive data and ensure it is stored securely.
- Remember the password that was used to encrypt this backup blob.

Restoring a Backup



Before restoring, you must first create a new Terraform Enterprise installation.

Once the Terraform Enterprise application is up and running, you can initiate a restore by making a POST request to the restore endpoint.

```
curl \
  --header "Authorization: Bearer $TOKEN" \
  --request POST \
  --form config=@payload.json \
  --form snapshot=@backup.blob \
  https://<TFE_HOSTNAME>/_backup/api/v1/restore
```

`config` is the password json used for taking the snapshot

Things to Know



- Be prepared to upload many gigabytes of data from the filesystem of whichever machine the request is sent from.
- For best performance and to avoid disconnections, we recommend sending this request from a server colocated with the Terraform Enterprise installation rather than from a workstation.
- Once the restore is complete, you must restart the application. The application will not restart automatically.

Reference links



- [Terraform Monitoring](#)
- [Terraform Logging](#)
- [Terraform Upgrades](#)
- [Terraform Backup and Restore](#)



Lab: When Things go Wrong

Begin and complete the entire track

<https://play.instruqt.com/hashicorp/tracks/tfe-breakfix-labs>

