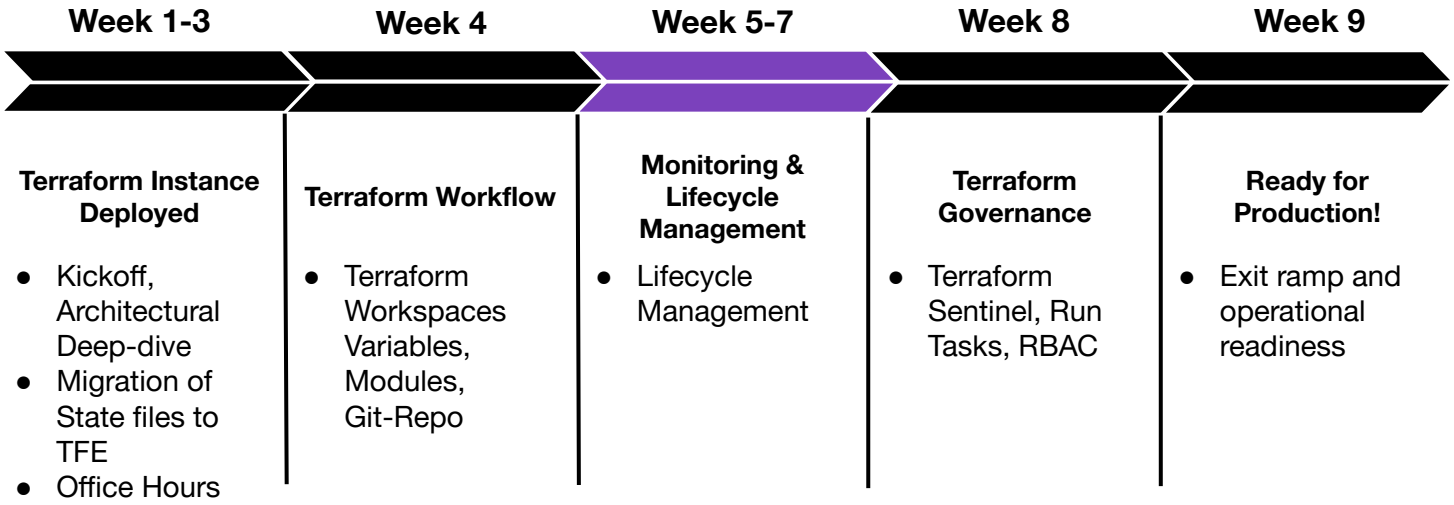
HashiCorp Terraform

Terraform Enterprise Lifecycle Management

Terraform Enterprise Path to Production



Agenda

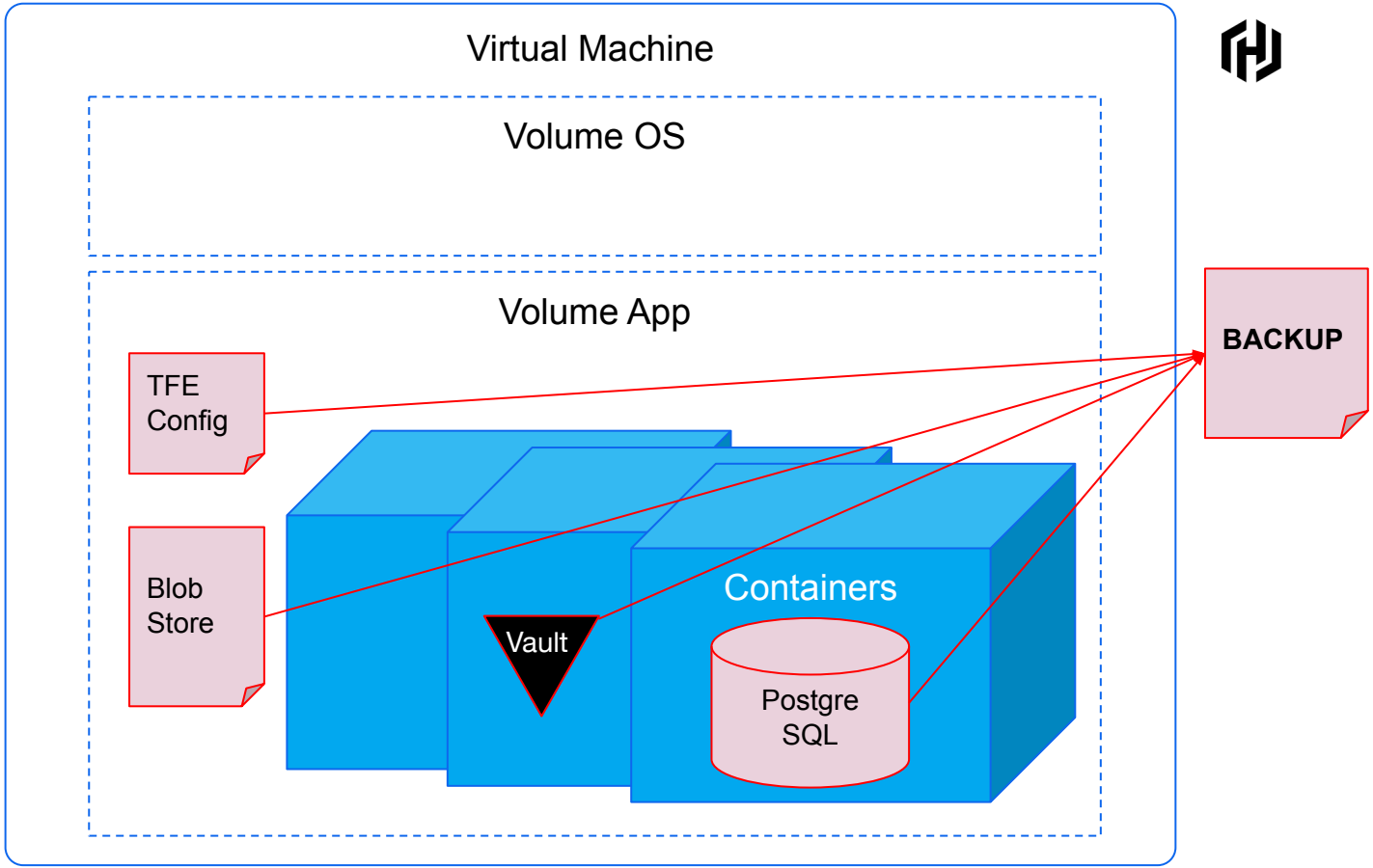
- Backup/Restore
- Upgrades
- Monitoring

Backup/Restore

Overview

A Terraform Enterprise (TFE) deployment is made up of a number of services including a Postgres database, object store, and Vault cluster. Depending on your operational mode, you will have varying levels of responsibility to ensure each component is backed up properly. TFE includes built in tooling to help backup the services required to run TFE.

What to Backup?



Backup and Restore

Responsibility Matrix

Your deployment mode will determine how much of your configuration can be captured using the built-in snapshot mechanism. In a production deployment you will be responsible for backing up PostgreSQL, Blob storage and potentially Vault.

	Configuration	Vault	PostgreSQL	Blob Storage
Mounted Disk	TFE*	TFE*	User	User
External Services	TFE*	TFE* (If an external Vault Cluster is used it is the users responsibility to backup)	User	User

*Once properly configured, automated snapshots will capture TFE server configuration and Vault configuration

Backup and Restore

Methods

1. VM Snapshot and restore.
   - The Snapshot needs to include RAM and should quiesce the database.
   - A maintenance window may be required for a full backup.
   - Fastest recovery.
2. Use the snapshot tool and separately backup db and blob storage.
   - The database should be quiesced.
   - A maintenance window may be required for a full backup.
3. Use the snapshot tool and the backup API.
   - No maintenance window required, but slower backup and restore process.

Backup and Restore

Blob Store and Database

1. Blob Storage is on the host VM, so just use your standard backup technology.
2. Postgre SQL is hosted in one of the containers, so use the replicated command line to back it up.

- replicated admin db-backup: This will run a pg_dump and store the backup in /backup/ptfe.db on the host.
- replicated admin db-restore: This will run a pg_restore using /backup/ptfe.db as it's data source.
- replicated admin db-reindex: This will run a REINDEX against the application database. Note: A reindex can take anywhere from minutes to hours to complete, depending on the size of your database. Running this command locks the database and prevents any other action against it.

Backup & Restore

Built in Methods

Automated Snapshots

Recommended automated snapshots.

Backup API

Terraform Enterprise API to backup and restore all application data.

Automated Snapshots

Mounted Disk Mode

- Automated snapshots are most effective when using mounted disk as the amount of backed up data is smaller and less risky.
- For Mounted Disk **daily is recommended** as the snapshot contains only configuration data.
- **Ensure to quiesce the database** on Mounted Disk instances — your backup software may or may not do this automatically.

Automated Recovery

Mounted Disk Mode

- Snapshot files can be used used to provision a new TFE instance using the latest snapshot file. Example scripts for restoring a snapshot to a new instance can be found here.
- For Airgap installations the license file and Airgap package must be on the instance prior to restore and in the same location as the original instance. The snapshot must also originate from an Airgap instance.

Backup API

Backup

Backup Terraform Enterprise with the new, dedicated Backup API to a device in a different region.

The API is separate from the main application-level APIs and uses a different token.

Use this method to migrate from Standalone to Clustered install types.

```
POST /_backup/api/v1/backup
```

Backup API

Restore

Use Backup API to restore your Terraform Enterprise platform.

As Backup backs up the database and the blob storage so a restore could contain gigabytes of data - use a co-located server for restores.

Restart the application after a restore using the UI or CLI.

```
POST /_backup/api/v1/restore
```

Backup & Restore

Best Practices

Ensuring TFE is operationally ready by:

- Hardening the service image using CIS benchmarks
- Run TFE on single-use machines
- Remove all unnecessary packages from the OS
- Deploy immutable instances
- Pin the version of TFE in the Replicated `install.sh` script

Maintaining the backup/restore process:

- Test the backup/restore process
- Document the backup/restore process
- Create and execute runbooks regularly

Upgrades

Overview

Terraform Enterprise (TFE), like any self-managed enterprise software, maintains a regular update cycle to ensure the stability and security of the application. TFE currently releases application updates on a monthly basis. These updates include application features, bug fixes, and security updates.

Upgrade Options

Immutable Virtual Machines vs In Place

Immutable Virtual Machines

- Reduce risk by enabling offline testing
- Reduce chance of creep in OS image standards and TFE configuration, all stored as code in Source Control
- A bit more initial overhead and skill required (e.g. Packer and cloud-init)
- Automated, consistent process

In Place

- OS patching and TFE updates are done in place
- High chance of drift from initial OS standards and TFE configuration over time
- Manual process with risk of human error
- Less skill required, easier for orgs with less mature IaC capabilities.

Upgrade Types

Immutable Virtual Machines

OS Update

- TFE Should be installed after the OS image is created.
- OS image just includes TFE dependencies and security hardening.
- TFE install should be run using cloud-init.
- Template can be tested using a copy of the TFE volume.

TFE Update

- TFE update should use the same process as OS Update, just update the versions of Replicated and / or TFE run by cloud-init.
- Testing is the same, take a copy of the TFE volume and run against that.

Upgrade High Level Steps

Immutable Virtual Machines

1. Run in dev / test environment first!
2. Update the VM image to target desired versions of Replicated and TFE.
3. Optionally, test offline using a volume snapshot
4. Schedule maintenance window
5. Stop access to TFE (e.g. take out of load balancer)
6. Take a Backup of 4 components (config, vault, blob, SQL)
7. Deploy the new image and let the update run against the live volume
8. Update the load balancer config to make the new instance available
9. Test
10. Repeat steps 4 to 9 in production environment

Upgrade High Level Steps

In Place

1. Run in ‘dev’ environment first!
2. Schedule maintenance window
3. Stop access to TFE (e.g. via load balancer)
4. Take a Backup of 4 components (config, vault, blob, SQL)
5. Run the TFE Update Process
6. Run the Replicated Update Process
7. Restore access to TFE
8. Test
9. Repeat steps 2 to 8 in production

Upgrading

Preparation

Preparing for upgrades in dev / test environments will ensure smoother Production upgrades.

Prior to upgrading a few key pre-req steps are helpful:

- Read the Terraform Enterprise Release Notes
- Export Terraform Enterprise Configuration
- Determine which releases are required

Upgrading

Preparation

Downtime is to be expected during upgrades.

Depending on whether database updates have to be applied anticipated downtime is 30 seconds to 30 minutes.

Ensure you **backup prior to upgrading** and verify the backup is viable.

Upgrade Process - UI

1. Navigate to `https://<TFE HOSTNAME>:8800/dashboard`
2. Click “Check Now”
3. Click “View Update”
4. Click “Install Update”
5. TFE will then automatically download and install the update. This usually takes 15-30 min depending on system and network performance

Upgrade Process - Airgapped - UI

1. Download airgap package to the update path location
   a. Update path can be found here: `https://<TFE HOSTNAME>:8800/console/settings`
2. Navigate to `https://<TFE HOSTNAME>:8800/dashboard`
3. Click “Check Now”
4. Click “View Update”
5. Click “Install Update”
6. TFE will then automatically download and install the update. This usually takes 15-30 min depending on system and network performance

Upgrade Process - CLI

1. Connect to the Terraform Enterprise host machine using SSH.
2. Fetch the versions of Terraform Enterprise.
   `replicatedctl app-release ls --fetch`
3. Upgrade to the latest version of Terraform Enterprise.
   `replicatedctl app-release apply`

   Alternatively, upgrade to a specific version of Terraform Enterprise.
   `replicatedctl app-release apply --sequence "504"`

Upgrade Process Replicated Components

1. Connect to the Terraform Enterprise host machine using SSH.
2. Re-run the Terraform Enterprise installation script for online installations
   `curl https://install.terraform.io/ptfe/stable | sudo bash`

Airgapped:

1. Download the latest airgapped components and from a shell on your instance, in the directory where you placed the latest.tar.gz installer bootstrapper:
   `tar xzf latest.tar.gz`
2. Re-run the Terraform Enterprise installation script for online installations
   `sudo ./install.sh airgap`

Upgrading

Troubleshooting

If issues occur during your TFE upgrade the Support Knowledge Base is an excellent resource, along with logging Support Tickets in the Support Portal.

Monitoring

Keeping TFE Up and Running

Monitoring

Overview

- Health Check Endpoint
- Metrics/Telemetry
- Internal Monitoring
- Log Forwarding
- Audit Logs

Monitoring

Health Check

TFE provides an external health check endpoint on each instance. If Terraform Enterprise is up, the health check will return a `200 OK`.

The health check endpoint is available at `/_health_check` and operates in 2 modes:

- Full check `/_health_check?full=1`
- Minimal check `/_health_check`

Monitoring

Health Check

Accessing Health Check external endpoint with curl:

```
 $ curl http://$(docker inspect ptfe_health_check|jq -r .
[].NetworkSettings.Networks[].IPAddress):23005/_health_check
```

Will return:


```
 {"passed":true,"checks":[{"name":"Archivist Health Check","passed":true},
{"name":"Terraform Enterprise Health Check","passed":true},{"name":"Terraform
Enterprise Vault Health Check","passed":true},{"name":"Fluent Bit Health
Check","passed":false,"skipped":true},{"name":"RabbitMQ Health
Check","passed":true},{"name":"Vault Server Health Check","passed":true}]}
```

Monitoring

Health Check

TFE provides an internal health check

The internal health check will verify the following:

- Archivist is up and healthy.
- The application can communicate with Redis and Postgres.
- The application can connect with Vault and is able to encrypt and decrypt tokens.
- Test that RabbitMQ is able to send and consume messages.
- Verify that the Vault server is healthy.

Monitoring

Health Check

Internal Health Check: `tfe-admin health-check`

```
$ tfe-admin health-check
```

```
 checking: Archivist Health Check...
|  checks that Archivist is up and healthy
|- ✓ PASS

checking: Terraform Enterprise Health Check...
|  checks that Terraform Enterprise is up and can communicate with Redis and
Postgres
|- ✓ PASS

checking: Terraform Enterprise Vault Health Check...
|  checks that Terraform Enterprise can connect to Vault and is able to encrypt and
decrypt tokens
|- ✓ PASS

checking: RabbitMQ Health Check...
|  checks that RabbitMQ can be connected to and that we can send and consume
messages
|- ✓ PASS

checking: Vault Server Health Check...
|  checks that the configured Vault Server is healthy
|- ✓ PASS

  All checks passed.
```

Monitoring

Metrics / Telemetry

Monitoring of standard server metrics is recommended:

- I/O
- CPU
- RAM
- Disk

Container Metrics:

- JSON
- Prometheus format

Monitoring

Metrics

Metadata labels will added to each container metric:

- `id` : The container ID
- `name` : The container name
- `Image` : The container image

Additional labels included with Build worker container metrics:

- `run_type`
- `run_id`
- `workspace_name`
- `organization_name`

Global Metrics:

- `tfe.run.count`
- `tfe.run.limit`

Monitoring

Container Metrics

Runtime Metrics

- tfe.container.cpu.usage.user
- tfe.container.cpu.usage.kernel
- tfe.container.memory.used_bytes
- tfe.container.memory.limit
- tfe.container.network.rx_bytes_total
- tfe.container.network.rx_packets_total
- tfe.container.network.tx_bytes_total
- tfe.container.network.tx_packets_total
- tfe.container.disk.io_op_read_total
- tfe.container.disk.io_op_write_total
- tfe.container.disk.io_bytes_read_total
- tfe.container.disk.io_bytes_write_total
- tfe.container.process_count
- tfe.container.process_limit

Monitoring

Metrics / Enable

Metrics collection can be configured with the config flag in the application config file.

`metrics_endpoint_enabled` Default is set to "0" (disabled). To enable metrics collection, set this value to "1".

`metrics_endpoint_port_http` Defines the TCP port on which HTTP metrics requests will be handled. Defaults to 9090

`metrics_endpoint_port_https` Defines the TCP port on which HTTPS metrics requests will be handled. Defaults to 9091

Access metrics in JSON: `/metrics`

Access metrics in Prometheus format:
`/metrics?format=prometheus`

Audit Logging

Audit Log entries contain `[Audit Log]` string:

```
2021-08-31 04:58:30 [INFO] [7a233ad1-c50c-4737-a925-3be901e55fcb] [Audit Log]
{
  "resource":"run",
  "action":"create",
  "resource_id":"run-nL77p69bsesoF3RK",
  "organization":"example-org",
  "organization_id":"org-pveSPvxocni226Fn",
  "actor":"example-user",
  "timestamp":"2021-08-31T04:58:30Z",
  "actor_ip":"19.115.231.192"
}
```

Log Forwarding

Requirements

- Terraform Enterprise running on an instance using `systemd-journald`
- A version of Docker that supports the `journald` logging driver
- Network connectivity between Terraform Enterprise and the external destination(s) where logs should be forwarded

Log Forwarding

Enable

- Log forwarding is disabled by default. To enable log forwarding, set the `log_forwarding_enabled` TFE application setting to the value 1.
- The `log_forwarding_config` TFE application setting must contain valid Fluent Bit `[OUTPUT]` configuration specifying supported external destination(s) where TFE should forward logs.
- Restart TFE

Log Forwarding

Supported External Destinations

- Amazon CloudWatch
- Amazon S3
- Azure Blob Storage
- Azure Log Analytics
- Datadog
- Forward
- Google Cloud Platform Cloud Logging
- Splunk Enterprise HTTP Event Collector (HEC)
- Syslog

Log Rotation

- To limit disk utilization of Log forwarding, configure the `SystemMaxFileSize` and `SystemMaxFiles` settings within `/etc/systemd/journald.conf`

```
[Journal]
SystemMaxFileSize=1024M
SystemMaxFiles=7
```

Next Steps

Need Additional Help?

Customer Success

Contact our Customer Success Management team with any questions. We will help coordinate the right resources for you to get your questions answered.

customer.success@hashicorp.com

Technical Support

Something not working quite right? Engage with HashiCorp Technical Support by opening a new ticket for your issue at support.hashicorp.com.

Upcoming Onboarding Webinars

May 24:

Week 6: **Terraform Workflows** (Modules, Workspaces, Git)

May 31:

Week 7: **Terraform Enterprise Arch Deep Dive** (Active/Active)

Community Office Hours

Resources

Backup/Restore

- TFE Backup and Restore
- TFE Backup (Learn Guide)
- Database Maintenance

Upgrades

- Upgrading TFE
- Availability During Upgrades
- Terraform Enterprise Release Notes
- Export Terraform Enterprise Configuration

Monitoring

- Health Check Endpoint
- TFE Metrics
- Internal Monitoring
- TFE Log Forwarding
- Audit Logs

Q & A

Thank You

customer.success@hashicorp.com
www.hashicorp.com