

Migrate from Vault OSS to Enterprise

Agenda

In-Place Migration 01

Storage Migration 02

Migration to New Vault Cluster 03

Automate Vault Configuration 04

01



In-Place Migration

Overview

1. The most common path for migrating an existing Vault Open Source cluster to Vault Enterprise is **via in-place migration**
2. In-place migration follows the [standard upgrade procedure](#) by simply replacing the existing Vault OSS binary with the Vault Enterprise version

In-Place Migration Process

1. Backup Vault cluster
2. Identify leader node
3. Replace binary on a follower node
4. Add licensing configuration to follower node
5. Repeat on all follower nodes
6. Replace binary and add licensing to leader node

Backup & Identify Leader Node

Consul Storage

```
1 $ consul operator raft snapshot
```

```
2 $ curl $VAULT_ADDR/v1/sys/leader
```


```
{  
  "ha_enabled": true,  
  "is_self": false,  
  "leader_address": "https://172.10.16.50:8200/",  
  "leader_cluster_address": "https://172.10.16.50:8201/",  
  "performance_standby": false,  
  "performance_standby_last_remote_wal": 0  
}
```

Backup & Identify Leader Node

Integrated Storage

```
1 $ vault operator raft snapshot save vault-oss.snapshot
```

```
2 $ vault operator raft list-peers
```

	Node	Address	State	Voter
	----	-----	-----	-----
	raft_node_1	172.10.16.50:8200	leader	true
	raft_node_2	172.10.16.53:8200	follower	true
	raft_node_3	172.10.16.56:8200	follower	true

Upgrade Binary on Follower Nodes

```
# Stop Vault on Follower node
$ systemctl stop vault

# Download ENT Binary
$ wget https://releases.hashicorp.com/vault/1.13.2+ent/vault_1.13.2+ent_linux_amd64.zip

# Replace existing Vault binary and then validate binary version
$ vault -v
Vault v1.13.2+ent (e9239da75f9f90570c20344d7b29900b1ed6eab416b38a2f6220a6f857cd662)

# STOP - Do not start Vault yet proceed to step 4 for licensing
```


Add Vault License on Follower Nodes

```
# 3 methods to autoload license, same method should be used on all nodes
```

1. Update `configuration file` with `license_path` parameter

```
License_path = "/ect/vault.d/license.hcllic"
```

2. Provide license path via environment `variable`

```
export VAULT_LICENSE_PATH = "/ect/vault.d/license.hcllic"
```

3. Provide license as a `string` in `environment variable`

```
export VAULT_LICENSE = "02MV4UU43BK5HGYTJZ..."
```

Start Vault on Follower Nodes



```
$ systemctl start vault
```

```
# Manually unseal node if not using an auto seal
```

```
$ vault operator unseal <unseal_key>
```

```
# Check Vault Status
```

```
$ vault status
```

```
# Verify logs are not outputting an errors
```

```
$ journalctl -u vault
```

```
# Repeat steps 1 - 5 on remaining follower nodes then proceed to step 6
```



Vault Enterprise

Repeat steps 1 - 5 on the leader node once all followers have been migrated successfully

02

Storage Migration

Vault Enterprise

Vault Enterprise supports two storage backends:

- **Integrated Storage**
- **Consul Storage**

If using an OSS supported storage backend you will need to migrate storage **prior** to upgrading to Vault Enterprise

Storage Migration

- Vault's “**operator migrate**” command copies data between storage backends
- Operates directly at the storage level, with no decryption involve
- Destination **should not** be initialized prior to the migrate operation
- Source data is not modified, except for a lock key added during migration
- This is an offline operation for data consistency, thus **requires downtime**

Storage Migration Process

1. Backup Vault Cluster
2. Create migration configuration file
3. Identify node to use for migration operation
4. Stop Vault
5. Run the migration
6. Update Vault configuration file(s)
7. Start and unseal Vault
8. Join additional nodes

Example Migration Configuration File

```
$ cat migrate.hcl

storage_source "mysql" {
  username = "user1234"
  password = "secret123!"
  database = "vault"
}

storage_destination "raft" {
  path = "/path/to/raft/data"
  node_id = "raft_node_1"
}

cluster_addr = "http://172.10.16.50:8200"
```


Migration Operation

```
$ vault operator migrate -config migrate.hcl
```

```
2018-09-20T14:23:23.656-0700 [INFO ] copied key: data/core/seal-config
```

```
2018-09-20T14:23:23.657-0700 [INFO ] copied key: data/core/wrapping/jwtkey
```

```
2018-09-20T14:23:23.658-0700 [INFO ] copied key: data/logical/fd1bed89-ffc4-d0
```

```
2018-09-20T14:23:23.660-0700 [INFO ] copied key: data/logical/fd1bed89-ffc4-d6
```

```
...
```

03

Migration to New Cluster

Overview

- The most typical pattern utilized is to perform an in-place migration to Vault Enterprise
- Some teams opt for a fresh start for their Vault Enterprise deployment
- Currently, Vault does not have built-in migration to move data from one Vault cluster to another
- Some elements of Vault migration can be automated using Vault's API and/or tooling developed by the community

Static Secrets

Export static secrets
from current cluster
and import from CSV

```
#!/bin/bash
set -e
COMMAND="vault kv put kv-v1/sample"
while IFS="," read -r key value
do
    COMMAND="$COMMAND $key=$value"
done < secrets.csv
eval $COMMAND
```

Policies

Export policies from
current cluster and
import from CSV

```
#!/bin/bash
{
  #ignores first line
  read -r
  while IFS="," read -r name file
  do
    vault policy write "$name" "$file"
  done
} < policy-names.csv
```

Transit Keys

Transit keys can only be exported if they had initially been created with exportable set to true

```
#Run against current cluster

#!/bin/bash
KEYS=$(vault list -format=json transit/keys | jq
.[] | sed 's/"//g')
for key in $KEYS
do
    vault write transit/keys/"$key"/config
allow_plaintext_backup=true exportable=true
    vault read -format=json transit/backup/"$key" | jq
.data > backups/"$key"-backup.json
done

#Run against new cluster

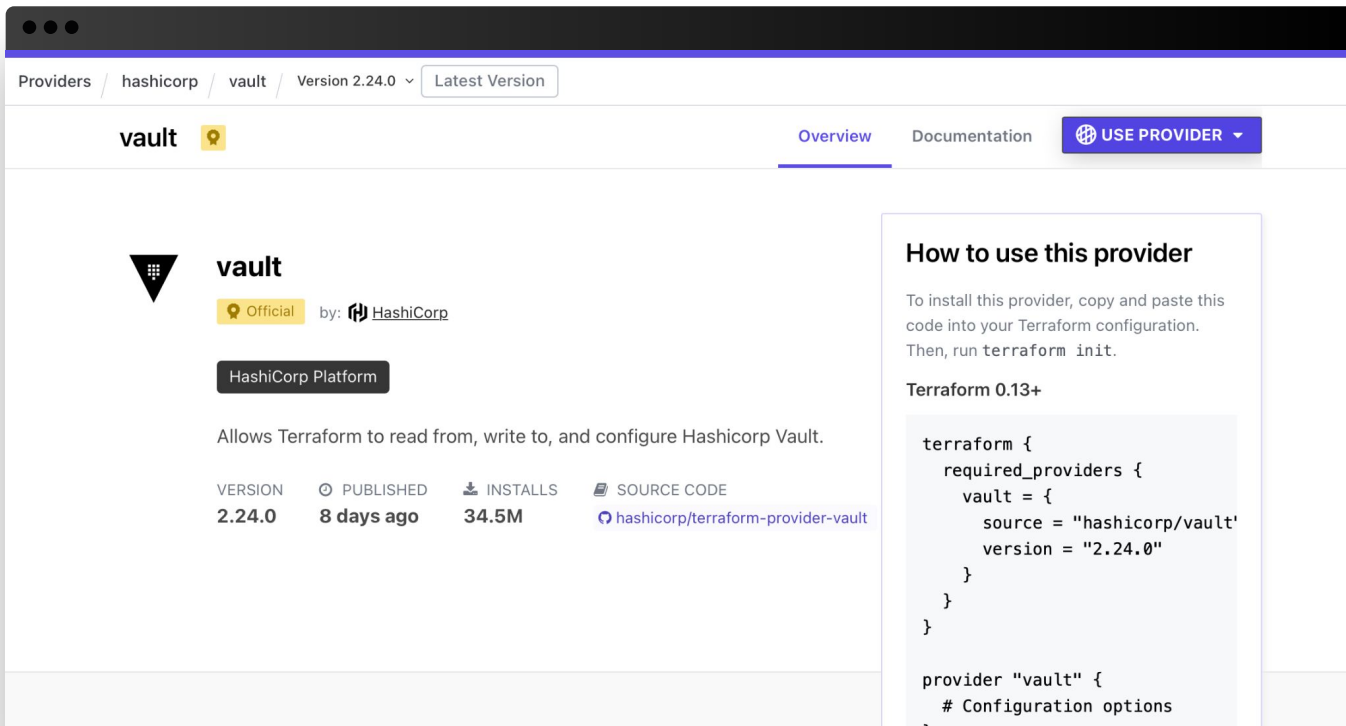
#!/bin/bash
for file in backups/*.json
do
    vault write transit/restore @"$file"
done
```

04

Automate Vault Configuration


Vault Provider

Provision namespaces, policies, secrets engines, and auth methods






The screenshot shows the HashiCorp Vault Provider page on Terraform Hub. The breadcrumb navigation at the top reads: Providers / hashicorp / vault / Version 2.24.0 (Latest Version). The page title is 'vault' with an official provider icon. Navigation links include 'Overview' (active), 'Documentation', and a 'USE PROVIDER' button. The provider details section shows the 'vault' icon, an 'Official' badge, and 'by: HashiCorp'. A 'HashiCorp Platform' badge is also present. The description states: 'Allows Terraform to read from, write to, and configure Hashicorp Vault.' Below this, a table lists the version (2.24.0), published date (8 days ago), install count (34.5M), and source code link (hashicorp/terraform-provider-vault). A 'How to use this provider' section provides instructions on installing the provider by copying code into a Terraform configuration and running 'terraform init'. It specifies 'Terraform 0.13+' and shows a code snippet for the provider configuration.

Providers / hashicorp / vault / Version 2.24.0 (Latest Version)




vault 

[Overview](#) [Documentation](#) [USE PROVIDER](#)

 **vault**  by:  HashiCorp

HashiCorp Platform

Allows Terraform to read from, write to, and configure Hashicorp Vault.

VERSION	 PUBLISHED	 INSTALLS	 SOURCE CODE
2.24.0	8 days ago	34.5M	hashicorp/terraform-provider-vault

How to use this provider

To install this provider, copy and paste this code into your Terraform configuration. Then, run `terraform init`.

Terraform 0.13+

```
terraform {
  required_providers {
    vault = {
      source = "hashicorp/vault"
      version = "2.24.0"
    }
  }
}
```

```
provider "vault" {
  # Configuration options
}
```


Namespace and Provider Alias

```
resource "vault_namespace" "infosec" {  
  path = "infosec"  
}  
  
provider vault {  
  alias      = "infosec"  
  namespace = vault_namespace.infosec.path  
}  
  
resource "vault_policy" "example" {  
  provider = vault.infosec  
  ...  
}
```

Create Policy

Create auth method
for OIDC provider

```
data "vault_policy_document" "dev_user_policy" {  
  rule {  
    path          = "secret/data/development/*"  
    capabilities = ["create", "read", "update", "delete",  
"list"]  
  }  
}  
  
resource "vault_policy" "devusers" {  
  name     = "dev-policy"  
  policy   = "${data.vault_policy_document.hcl}"  
}
```

Enable User Auth Method

Create auth method for
OIDC provider

```
resource "vault_jwt_auth_backend" "oidcauth" {  
  description = "Auth0 OIDC"  
  path        = "oidc"  
  type        = "oidc"  
  oidc_discovery_url = "https://myco.auth0.com/"  
  oidc_client_id  = "1234567890"  
  oidc_client_secret = "secret123456"  
  bound_issuer    = "https://myco.auth0.com/"  
  tune {  
    listing_visibility = "unauth"  
  }  
}
```

Create Auth Role

Role will define the user claim to authenticate a user and which policy assignments they have in Vault

```
resource "vault_jwt_auth_backend_role" "example" {  
  backend      = vault_jwt_auth_backend.oidc.path  
  role_name    = "test-role"  
  token_policies = ["default", "dev", "prod"]  
  
  user_claim      = "https://vault/user"  
  role_type       = "oidc"  
  allowed_redirect_uris =  
["http://localhost:8200/ui/vault/auth/oidc/oidc/callback"]  
}
```

Enable Secrets Engines

```
resource "vault_mount" "kv2-infosec" {  
  path      = "infosec"  
  type      = "kv-v2"  
}  
  
resource "vault_mount" "pki-dev" {  
  path      = "pki-dev"  
  type      = "pki"  
  default_lease_ttl_seconds = 3600  
  max_lease_ttl_seconds    = 86400  
}
```

Best Practices

Protect State

- Terraform, by default, stores state in the working directory where Terraform CLI is executed
- Remote State should be used and encrypted
- Access to state should be limited by following practice of least privilege

Manage as Code

- Treat Terraform configuration files as code & store in VCS
- Practice least privilege for access and who can commit changes
- Integrate into CI process and ensure code is tested in dev before pushing to production

Sensitive Values

- Do not put any secrets in code
- Pass any secrets, such as credentials or Vault token by using environment variables
- Sensitive values may appear in state if not handled correctly

Resources

- [Vault Upgrade Standard Procedure](#)
- [Vault Data Backup Standard Procedure](#)
- [Upgrading Vault - Guides](#)
- [Operator Migrate](#)
- [License Autoloading](#)
- [Vault Provider for Terraform](#)
- [Migration Strategies and Considerations](#)
- [Related Tools](#)

Q&A



Thank you

customer.success@hashicorp.com

www.hashicorp.com/customer-success