KPLABS Course

HashiCorp Certified: Vault Associate

Vault Architecture

ISSUED BY
Zeal

REPRESENTATIVE
instructors@kplabs.in

Module 1: Vault for Production Environments

Till now, we have been using Vault under development mode where all data was stored in memory.

For production, we need a better storage class to store the data.

There are multiple storage class that can be used, some of these includes:

- Filesystem
- S3
- Consul
- Databases (MySQL, DynamoDB, PostgreSQL)

Step 1 - Create a configuration file

Outside of development mode, Vault servers are configured using a file.

The format of this file is HCL or JSON

```
storage "file" {
  path  = "/root/vault-data"
}

listener "tcp" {
 address     = "0.0.0.0:8200"
 tls_disable = 1
}
```

.

Step 2 - Start Vault Server from Configuration File

Start the vault server by specifying the configuration file created in the first step.

```
[root@ip-172-31-87-159 ~]# vault server -config demo.hcl
==> Vault server configuration:

                     Cgo: disabled
              Listener 1: tcp (addr: "0.0.0.0:8200", cluster address: "0.0.0.0:8201",
 max_request_size: "33554432", tls: "disabled")
               Log Level: info
                   Mlock: supported: true, enabled: true
           Recovery Mode: false
                 Storage: file
                 Version: Vault v1.4.1

==> Vault server started! Log data will stream in below:
```

Step 3 - Initialize Vault

This only happens once when the server is started against a new backend that has never been used with Vault before.

During initialization, the encryption keys are generated, unseal keys are created, and the initial root token is set up.

```
[root@ip-172-31-87-159 ~]# vault operator init
Unseal Key 1: JbJ21oVoUH+WGA7zm9oiRnyDdBpku8lpBgyZAnNH7LOg
Unseal Key 2: 5pLbx/hdfHU9pXuwGzTwk+7zfEQqgc52/4J8qDOT7Y2f
Unseal Key 3: rcQQuUFl2C+fiySkS9HEWGJ8WyyOMussqVoX4CxSJX/+
Unseal Key 4: Nb/clnrQNyOP2OJezJJRgJ/IbL+H1RdTjZJbgDa+vRUv
Unseal Key 5: 9gHS2DD4IqrgBR4+LskmJXMoF0a0nLVvHfe/p9yW6KOV

Initial Root Token: s.hAkkaVU0e4NKyzLaPewKIPfX
```

,

Step 4 - Unseal the Vault

Every initialized Vault server starts in the sealed state. From the configuration, Vault can access the physical storage, but it can't read any of it because it doesn't know how to decrypt it.

The process of teaching Vault how to decrypt the data is known as unsealing the Vault.

```
[root@ip-172-31-87-159 ~]# vault operator unseal
Unseal Key (will be hidden):
Key                Value
---                -----
Seal Type          shamir
Initialized        true
Sealed             true
Total Shares       5
Threshold          3
Unseal Progress    1/3
Unseal Nonce       c574f8ba-a5c5-4877-fbe8-52cef9ef42b3
Version            1.4.1
HA Enabled         false
```

Important Pointers - Initialization State

Initialization outputs two incredibly important pieces of information: the unseal keys and the initial root token.

This is the only time ever that all of this data is known by Vault, and also the only time that the unseal keys should ever be so close together.

Module 2: Vault UI For Production

Vault features a user interface (web interface) for interacting with Vault.

The Vault UI is not activated by default. To activate the UI, set the UI configuration option in the Vault server configuration. Vault clients do not need to set this option since they will not be serving the UI.

You can enable Vault UI by adding the following parameter within the configuration file:

ui = true

Module 3: Vault Agent

3.1 Overview of Challenge

For the application that needs to interact with Vault, it first needs to authenticate and then use the tokens for performing required tasks.

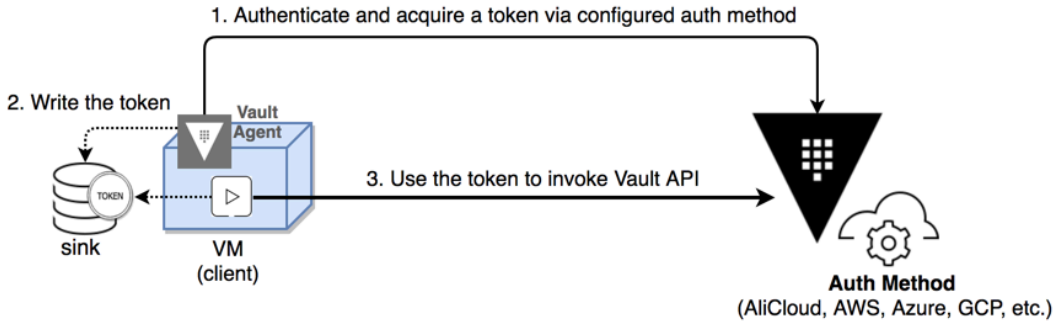
Apart from this, the application also needs to have logic related to token renewal and others.

Instead of building custom logic in the application, you can instead make use of Vault agent.

3.2 Overview of Vault Agent

Vault agent is a client daemon that automates the workflow of client login and token refresh.

- Automatically authenticates to Vault for those supported auth methods
- Keeps token renewed (re-authenticates as needed) until the renewal is no longer allowed
- Designed with robustness and fault tolerance

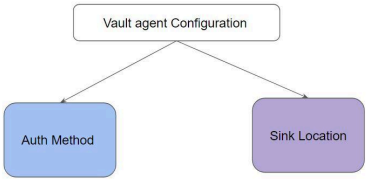


3.3 Running Vault Agent

In order to make use of Vault agent, you can run the vault binary in the agent mode

vault agent config=<config-file>

The agent configuration file must specify the auth method and sink locations where the token to be written.



3.4 Working of Vault Agent

When the agent is started, it will attempt to acquire a Vault token using the auth method specified in the agent configuration file.

On successful authentication, the resulting token is written to the sink locations.

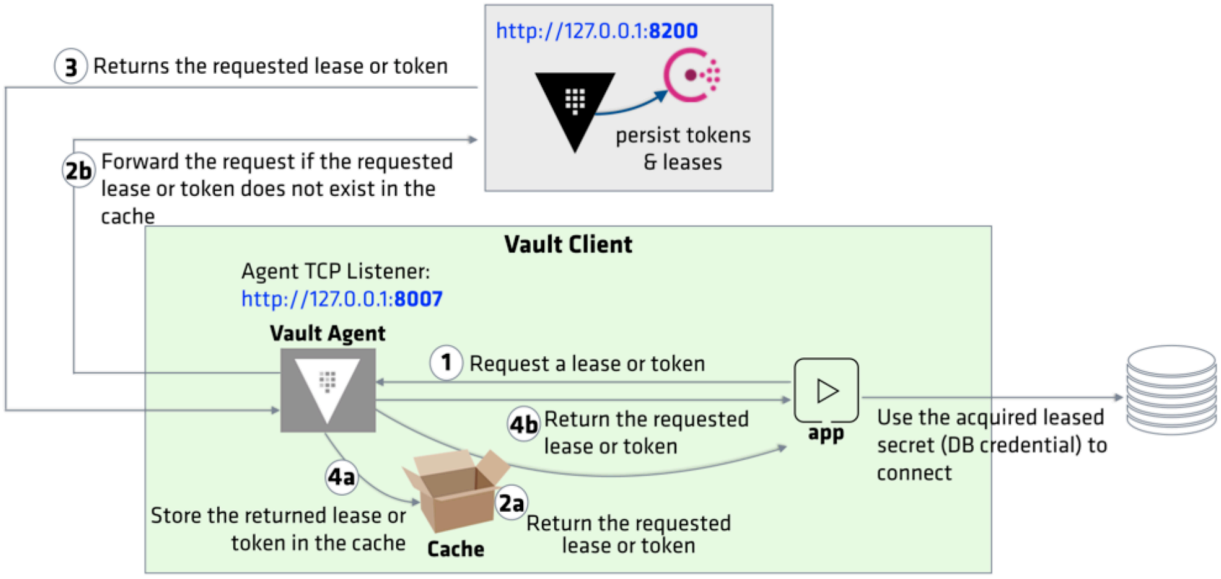
Whenever the current token value changes, the agent writes to the sinks.

Module 4: Vault Agent Caching

There are two primary functionalities related to Vault Agent:

Functionalities	Description
Auto-Auth	Automatically authenticate to Vault and manage the token renewal process.
Caching	Allows client-side caching of responses containing newly created tokens.

Vault Caching feature allows client-side caching of responses containing newly created tokens and responses containing leased secrets generated off of these newly created tokens.



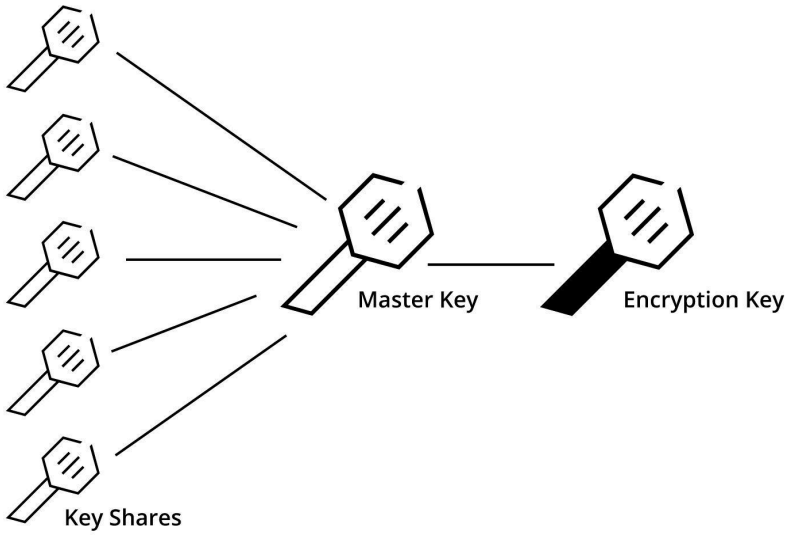
Module 5: Shamir Secret Sharing

5.1 Overview of the Basics

The storage backend in Vault is considered to be untrusted. The data stored in the storage backend is within the encrypted state.

When the Vault is initialized it generates an encryption key which is used to protect all the data. That key is protected by a master key.

Vault uses a technique known as Shamir's secret sharing algorithm to split the master key into 5 shares, any 3 of which are required to reconstruct the master key.



5.2 Important Pointers

The number of shares and the minimum threshold required can both be specified.

Shamir's technique can be disabled, and the master key used directly for unsealing.

Once Vault retrieves the encryption key, it is able to decrypt the data in the storage backend and enters the unsealed state.

5.3 Seal Stanza

The seal stanza configures the seal type to use for additional data protection, such as using HSM or Cloud KMS solutions to encrypt and decrypt the master key.

This stanza is optional, and in the case of the master key, Vault will use the Shamir algorithm to cryptographically split the master key if this is not configured.

Module 6: Auto Unseal

5.1 Understanding the Challenge

During the vault unseal process, users who have the key need to enter them.

This allows each shard of the master key to be on a distinct machine for better security.

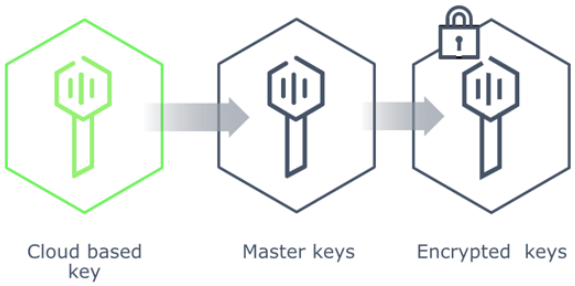
If there are multiple Vault clusters in the organization, unsealing can be a tiresome process were the availability of the users with keys needs to be present.

Unsealing makes the process of automating a Vault install difficult. Automated tools can easily install, configure, and start Vault, but unsealing it using Shamir is a very manual process.

5.2 Overview of Auto Unseal

Auto Unseal delegates the responsibility of securing the unseal key from users to a trusted device or service

At startup, Vault will connect to the device or service implementing the seal and ask it to decrypt the master key Vault read from storage.



5.3 Auto Unseal Options

There are various ways in which Vault can automatically be unsealed.

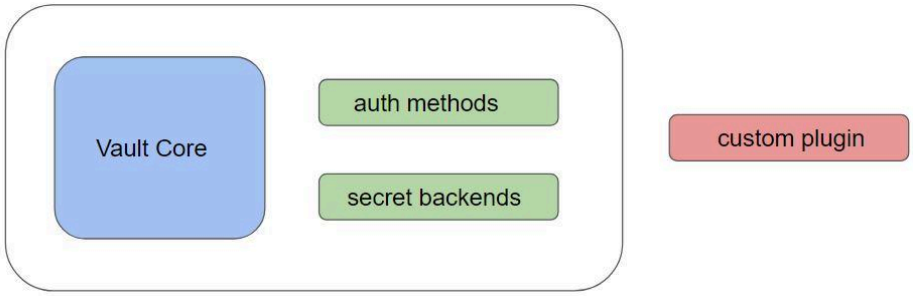
Following are some of the options:

- AWS KMS
- Transit Secret Engine
- Azure Key Vault
- HSM
- GCP Cloud KMS

Module 7: Vault Plugins

All Vault auth and secret backends are considered plugins.

This simple concept allows both built-in and external plugins to be treated like Legos.



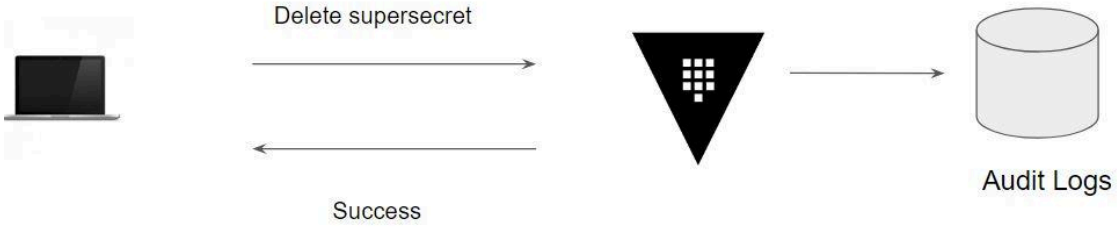
Following are the steps to integrate 3rd party plugins

1. Create a plugin
2. Compile the plugin.
3. Start Vault Server with an appropriate plugin directory path.

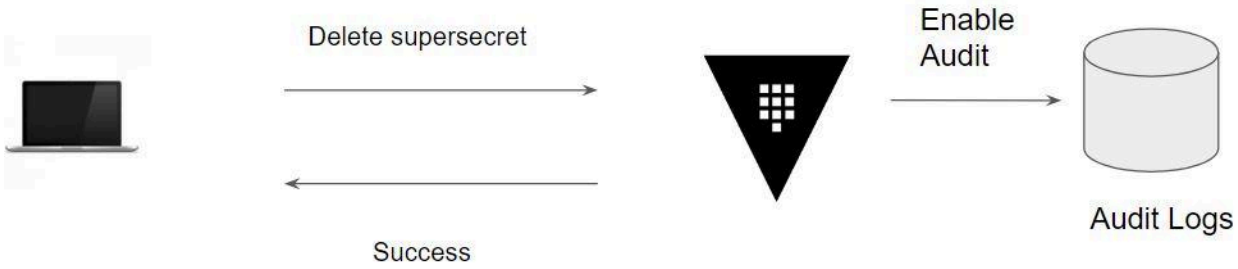
Module 8: Audit Devices

Audit devices are the components in Vault that keep a detailed log of all requests and responses to Vault.

Because every operation with Vault is an API request/response, the audit log contains every authenticated interaction with Vault, including errors.



When a Vault server is first initialized, no auditing is enabled. Audit devices must be enabled by a root user using vault audit enable



Important Pointers:

If there are any audit devices enabled, Vault requires that at least one be able to persist the log before completing a Vault request.

If you have only one audit device enabled, and it is blocking (network block, etc.), then Vault will be unresponsive. Vault will not complete any requests until the audit device can write.

If you have more than one audit device, then Vault will complete the request as long as one audit device persists the log.

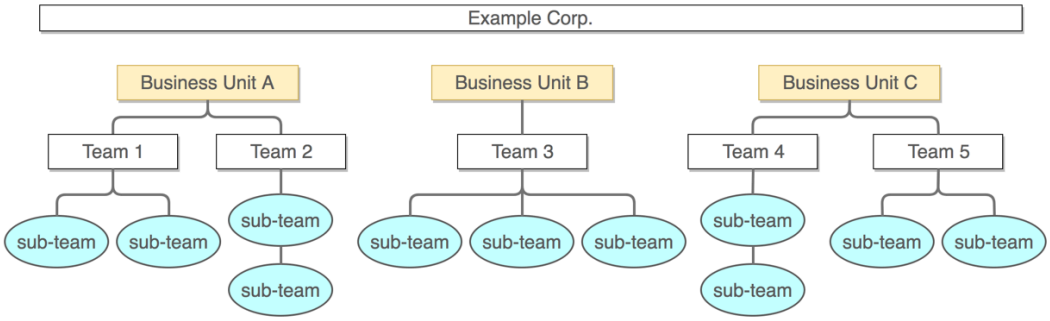
Module 9: Vault Namespaces

9.1 Understanding the Challenge

There can be multiple teams within organizations that might want to use Vault for their day-to-day operations in a self-serving manner.

Example: SysOps Team, Security Team.

Each team would like to manage their own secrets, auth methods, and others.

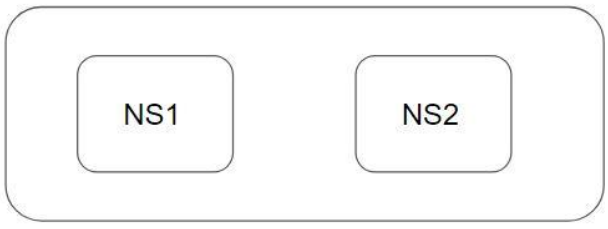


9.2 Overview of Namespace

Create a namespace dedicated to each team, organization, or app where they can perform all necessary tasks within their tenant namespace.

Each namespace can have its own:

- Policies
- Auth Methods
- Secrets Engines
- Tokens
- Identity entities and groups

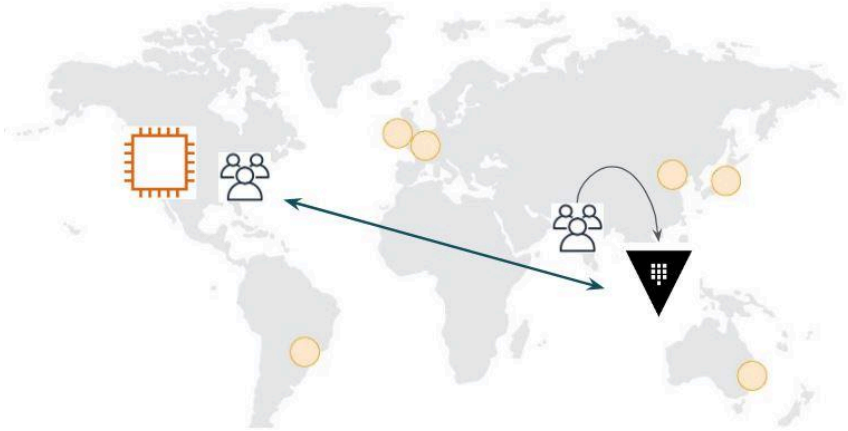


Vault within Vault

Module 10: Vault Replication

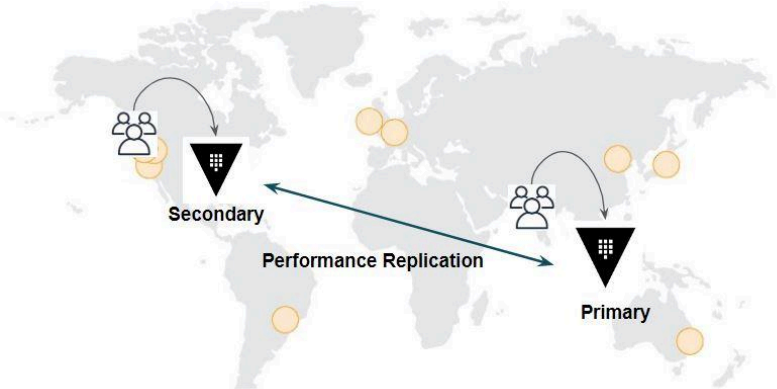
10.1 Understanding the Challenge

A single vault cluster can impose various challenges related to high latency, connectivity failures, availability loss, and others.



10.2 Overview of Performance Replication

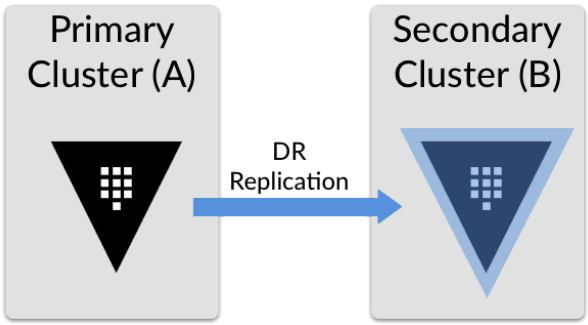
In performance replication, secondaries keep track of their own tokens and leases but share the underlying configuration, policies, and supporting secrets (K/V values, encryption keys for transit, etc).

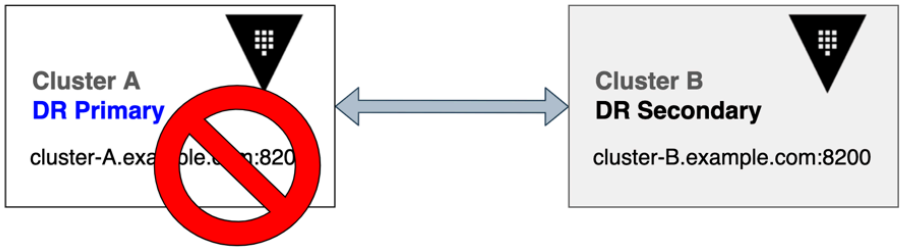


10.3 Overview of Disaster Recovery Replication

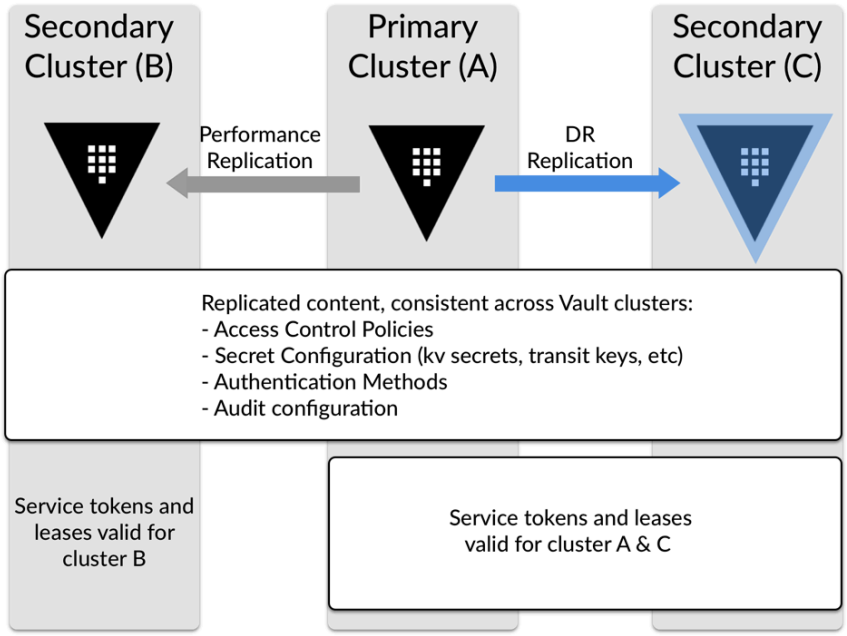
Ability to fully restore all types of data (local and the cluster data)

The secondary cluster does not handle any client requests and can be promoted to be the new primary in case of disaster.





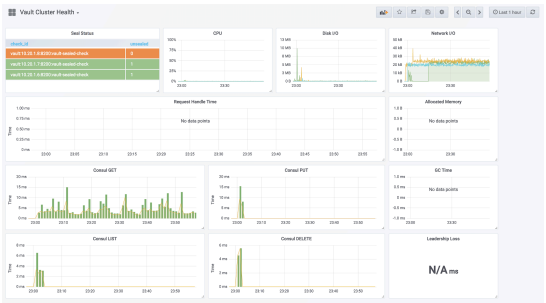
10.4 Performance Replication vs DR Replication



Module 11: Monitoring Telemetry

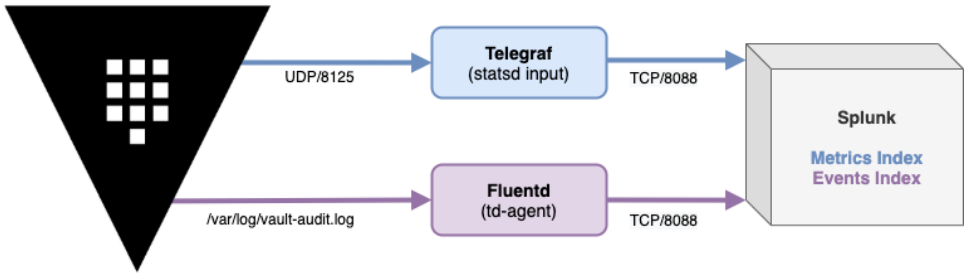
Telemetry describes the datasets that are coming out from a specific device.

This allows us to collect measurements to generate insights.



In the overall Vault monitoring, there are two important data sets that need to be collected:

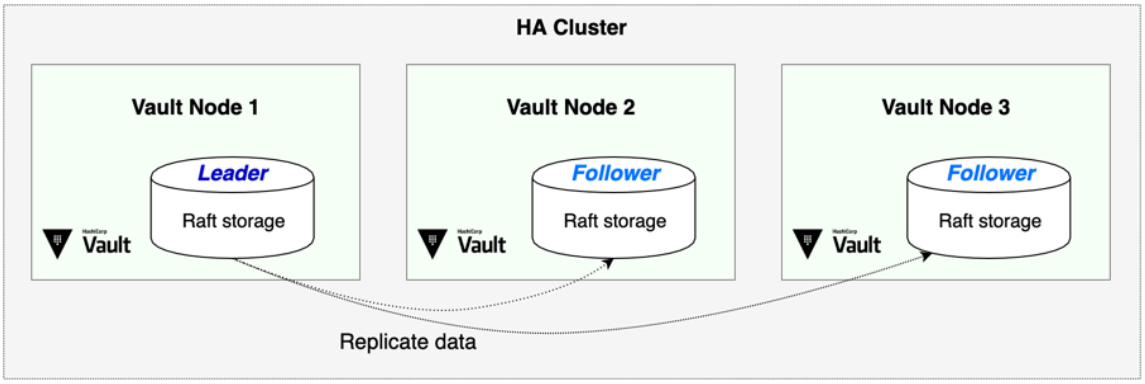
- Metrics
- Vault Audit Logs



Module 12: High-Availability Setup of Vault

Running a single Vault instance in production is risky.

Vault supports a multi-server mode for high availability. This mode protects against outages by running multiple Vault servers



To be highly available, one of the Vault server nodes grabs a lock within the data store.

The successful server node then becomes the active node; all other nodes become standby nodes.

At this point, if the standby nodes receive a request, they will either forward the request or redirect the client depending on the configuration.

The operator step-down forces the Vault server at the given address to step down from active duty

Module 13: Raft Storage - Snapshot and Restore

vault operator raft snapshot command allows us to perform snapshot and restore operation.

Command	Description
vault operator raft snapshot save demo.snap	Takes snapshot of vault data.
vault operator raft snapshot restore demo.snap	Restores a snapshot of Vault data.