Important points for
Hashicorp certified : Vault
Associate Exams

Dynamic secrets:

* Dynamic secrets allow you to generate credentials on-demand dynamically and are automatically revoked after a certain duration
* Not all the secret engine supports dynamic credentials
* Common supported Engines: AWS, Database, Google Cloud, Azure
* Dynamic secret does not provide any stronger cryptographic key generation

Lease Management:

* With every dynamic secret and service type authentication token, vault creates a lease: metadata containing informations such as a time duration, renewability and more
* Once the lease is expired, vault can automatically revoke the data, and the consumer of the secret can no longer be certain that it is valid

Lease options	Descriptions
Renew	This command renews the lease on a secret, extending the time that it can be used before it is revoked by vault
Revoke	When a lease is revoked, it invalidates that secret immediately and prevents any further renewals

vault lease renew -increment=3600 my-lease-id
(would request that the TTL of the lease be adjusted to 1 hour (3600 seconds))

vault lease revoke my-lease-id
(Revoke a specific lease)

vault revoke lease -prefix aws/
(Revoke all AWS access keys)

Transit Secrets Engine:

* The transit secrets engine handles cryptographic functions on data-in-transit
* All plaintext data must be base64-encoded
* The reason for this requirement is that vault does not require that the plaintext is "text". It could be a binary file such as a PDF or image
* we can rotate encryption keys at regular intervals to ensure that not all data is encrypted with just 1 encryption key

Transit Secrets Engine - key versions:

* The Transit engine supports the versioning of keys
* key versions that are earlier that a key's specified min_decryption_version get archived and rest of the key versions belong to the working set
* This leads to both performance benefits as well as security benefits
* By disallowing decryption of old versions of keys found ciphertext to obsolete (but sensitive) data can not be decrypted, but in an emergency, the

min_decryption_version can be moved back to allow for legitimate decryptions.

Vault policies:

* Vault policies are used to govern the access of users and roles (authorization)
* When you first initialize vault, the root and default policy gets created by default
* Policies are denied by default so an empty policy grants no permission in the system

```
path "sys/mounts" {
    capabilities = ["read"]
}
```

Default policy:

* The default policy is a built-in vault policy that cannot be removed
* By default, it is attached to all tokens, but can be explicitly excluded at token creation time by supporting authentication methods
* The policy contains basic functionality such as the ability for the token to lookup data about itself and to use its cubbyhole data
* However vault is not perspective about its contents. It can be modified to suit your needs.

ROOT policy:

* The root policy is a built-in vault policy that can not be modified or removed.
* A root user can do anything within the vault. As such as it's highly recommended that you revoke

any root tokens before running vault in production.
* When a vault server is first initialized, there already exists one root user. This user is used to do the initial configuration and setup of vault.
* After configured, the initial root token should be revoked and more strictly controlled users and authentication should be used.

Token Accessor:

This accessor is a value that acts as a reference to a token and can only be used to perform limited actions:

* Lookup a token's properties (not including the actual token ID)
* Lookup a token's capabilities on a path
* Renew a token
* Revoke the token

Policy Association:

* During the token creation time, the policy is associated with the token.
* At the later stage, if you attach a new policy to the user (or) role, it will not affect the existing tokens.
* For such cases, a new token must be created
* A policy which is updated and is already attached to the token, the rules will be reflected accordingly as part of the token's permission

Token Capabilities:

* The token capabilities command fetches the capabilities of a token for a given path.
* Token Accessor can also be used to check the capabilities of a specific token

Vault token capabilities secret/foo

O/p : read

Authentication Methods:

* There are multiple authentication methods available in Vault.
* After successful authentication, a token is generated which can be used for ~~authenti~~ interaction with Vault.
* Remember that the Github auth method is a user oriented method and is easiest to use for developer's machine
* For Servers, AppRole method is the recommended Choice

Vault auth enable -path=my-login userpass

Disabling Auth Method:

* A specific authentication method can be disabled with the following command:

Vault auth disable <method_name>

* When an auth method is disabled, all users authenticated via that methods are automatically logged out.

Storage Backend:

* Vault has multiple storage backends, each for different use-cases.
* Storage backends represents the location for the durable storage of vault's information
* Remember that not all the storage backends supports high-availability.

```
storage "dynamodb" {
  ha_enabled = "true"
  region     = "us-west-2"
  table      = "vault-data"
}
```

Disabling Secret Engine:

* The secrets disable command disables a secret engine at a given PATH
* Once a secrets engine is disabled, all secrets generated via secrets engine are immediately revoked.

```
vault secrets disable aws/
```

Login based on Userpass Auth Method:

While logging in via userpass auth method, it is important to not define the password directly within the CLI command.

```
vault login -method=userpass username=demo
                                      user
```

Unseal Vault:

* When a vault server is started, it starts in a sealed state.
* Unsealing is the process of constructing the master key necessary to read the decryption key to decrypt the data, allowing access to the vault.
* Unsealing is the process of reconstructing this master key

Vault Agent:

The vault agent does not persist anything to storage. Everything lives in memory.

There are two primary functionalities related to Vault Agent.

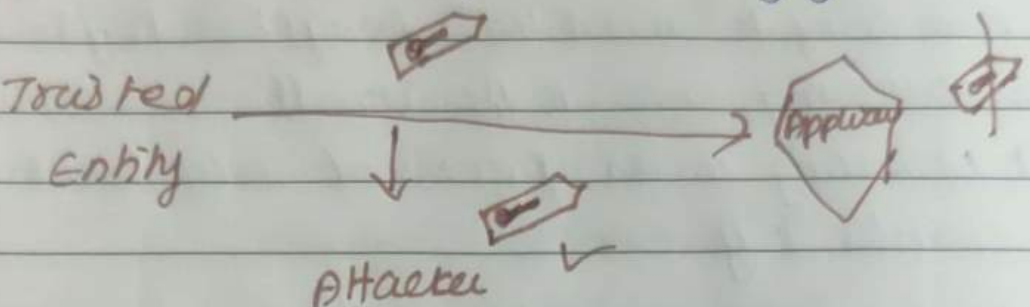
Auto-Auth → Automatically authenticate to vault and manage the token renewal process

Caching → * Allows client-side caching of responses containing newly created tokens.
* if configured with use_auto_auth_token, client will not be required to provide a vault token to the requests made to the Agent.

Response Wrapping Token:

* When response wrapping is requested, vault creates a temporary single-use token (wrapping token) and inserts the response into the token's cubbyhole with a short TTL

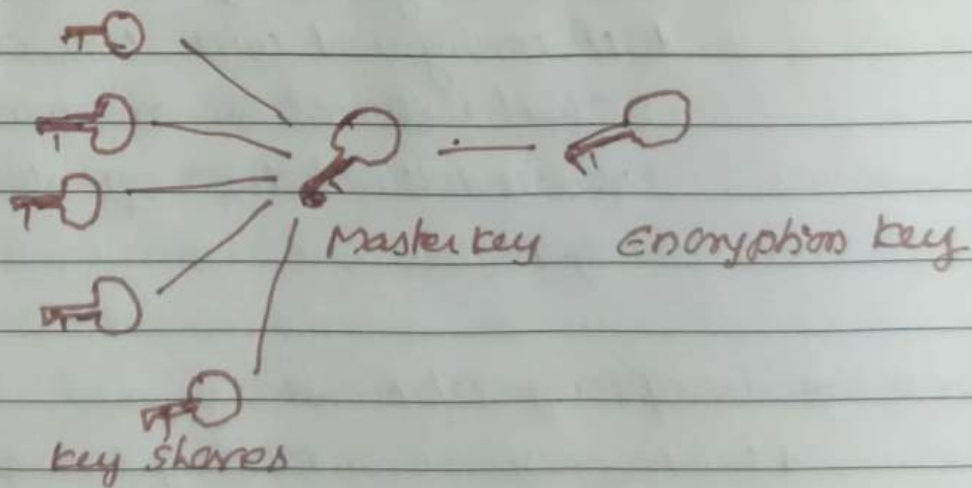
* If the wrapping token is compromised and attacker unwraps the secret, the application will not be able to unwrap again and this can sound an alarm and you can revoke things accordingly



Shamir Secret sharing for Unsealing Vault:

* The storage backend is considered to be untrusted and vault uses 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 of 3 which are required to reconstruct the master key.



* The best practise states that the key shares that are entered to get the master key should be done from a different workstation by different lessees having an individual key

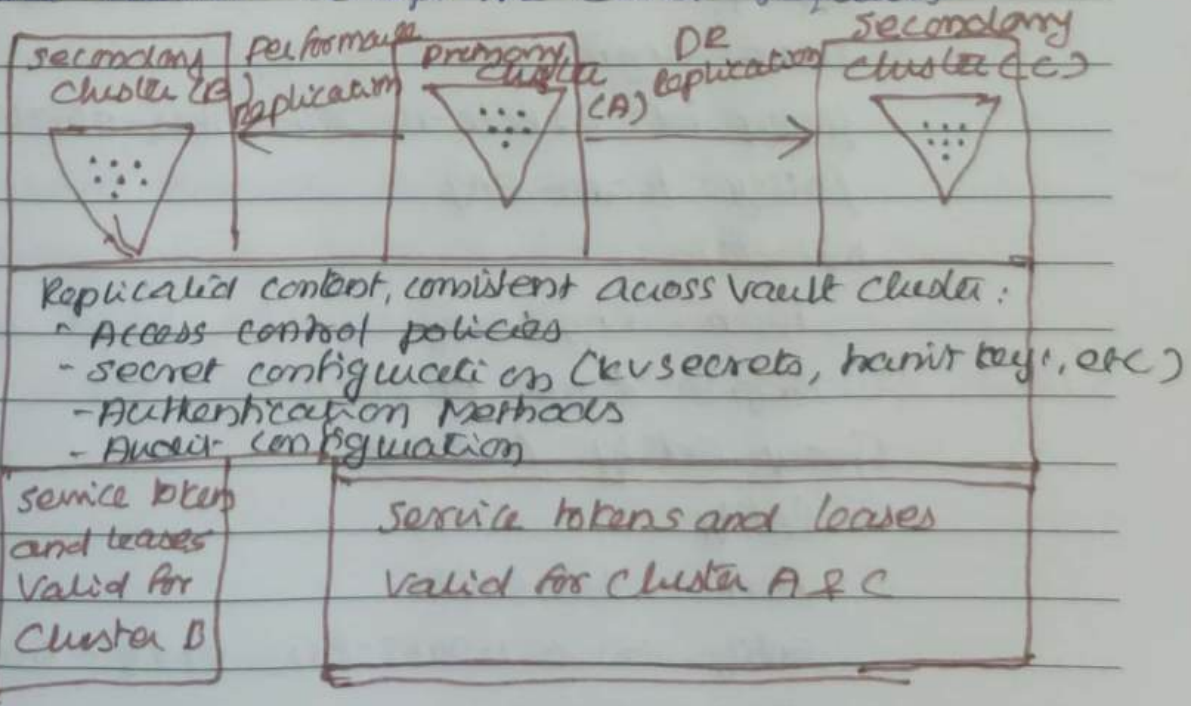
Seal Stanza:

* 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.

```
seal [NAME] {
  #....
}
```

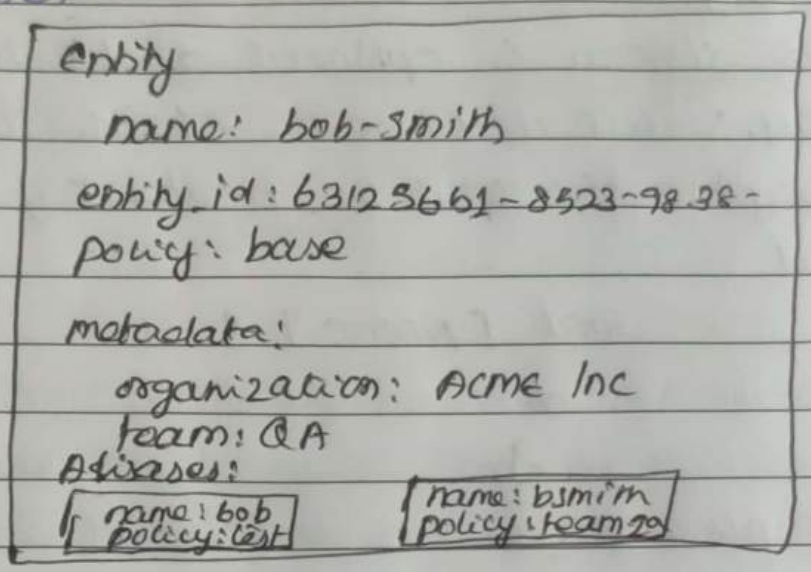
Vault Replication:

* For performance replication, secondary cluster will service reads locally.
* Some data is also stored locally and not replicated to the primary cluster
* In DR, all the data is replicated but the secondary cluster cannot accept the client requests



Entities and Aliases:

* Each client is internally termed as an entity. An entity can have multiple aliases.
* The policy defined at entity level are associated with the aliases



Identity Groups:

* A group can contain multiple entities as its members.
* policies set on the group is granted to all members of the group

Group
name: engineers
group_id: 81b0c90-284a-7b8c-9fa7b93bc
policy: team-eng
metadata:
team: Engineering
region: North America

Group Entity Member
Entity
name: Bob Smith
Entity_id: 6312 5661-8523-9838-5501
Policy: base

metadata:

organization: ACME Inc

team: QA

Aliases:

name: bob	name: bsmith
policy: test	policy: team-qa

Vault output:

We can have vault output in the following mode: Table, JSON, YAML

The default is the Table

Table View → vault secrets list -format table

JSON View → vault secrets list -format json

Multiple Encryption key:

* For Transit Engine, it is considered as a best practice to regularly rotate the encryption key.
* This limits the number of data encrypted via a single key
* All the data should not be encrypted with the single encryption key.
* The above increases the risk.

Reading output from kv path:

You want to be abled to read a specific secret at secret/demosecret. which capability needs to be used;

There are two primary capabilities:

* LIST
* READ

* List allows listing values at the given path.
* Read allows reading the data at the given path.

Audit Devices:

* Audit devices are the components in vault that keep a detailed log of all requests and responses to vault.
* When a vault server is first initialized, no auditing is enabled.
* Audit devices must be enabled by a root user using vault audit enable.

Vault Browser CLI:

* Allows running of basic CLI commands like write, read, delete and list.
* You will be not be able to perform various operations like creating new authentication methods and others

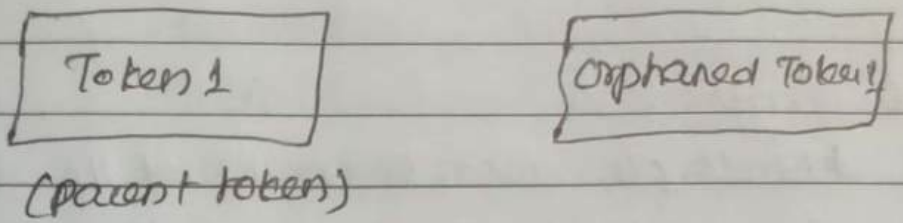
Vault token lookup:

vault token lookup s.G18DHQ5mnDNo6z53wTSbob(

It will show creation ttl, orphan and ttl.

Orphan Tokens:

* Orphan tokens are not children of their parents; therefore orphan tokens do not expire when their parent does.
* They are the root of their own token tree
* Orphan tokens still expire when their own max TTL is reached



Create a token with Explicit TTL:

* TTL = Initial TTL to associate with the token
* Explicit Max TTL: Maximum lifetime for the token
* Hard limits and cannot be exceeded
* The system max TTL, which is 32 days but can be changed in the vault's configuration file

```
Vault token create --ttl=100 --explicit-max-ttl=60
```

Renewing a Token:

Vault token renew command can be used to extend the validity of the renewable tokens.

```
Vault token renew --increment=30m
```

s.YcHxFMgNqT76O6iEE

Basic Environment Variables:

VAULT_ADDR

Address of the vault server expressed as a URL and port, for example, https://127.0.0.1:8200/ http://127.0.0.1:8200 can also be replaced with http://:8200

export VAULT_ADDR='http://:8200'

Secrets Engine:

* Multiple secrets engine of the same type can be enabled at a given time.
* We can distinguish them uniquely by separating them by the path
  - kv secret engine mounted on /secret
  - kv secret engine mounted on /kv

Supported Backend - Hashicorp Support

* Following BackEnds are officially supported by Hashicorp
* We will receive technical support for these backends.

In-Memory
File system
Consul
Raft

Vault Enterprise Features

Vault enterprise includes a number of features that may be useful in specific workflows.

It is important for us to understand the unique features that are part of Vault Enterprise

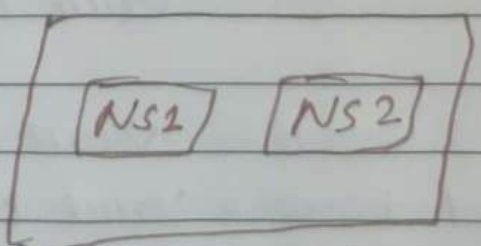
- Disaster Recovery
- Namespaces
- Monitoring
- Multi-Factor Authentication
- Auto-unseal with HSM

Vault Namespaces:

* Namespaces are isolated environments that functionally exist as "vaults within a vault"
* They have separate login paths and support creating and managing data isolated to their namespaces.

Each namespaces has can have its own:

- policies
- Auth Methods
- secrets Engines
- Tokens
- Identity entities and groups



Vault within vault

Vault Replication:

* Whenever you have replication enabled all of the secondary clusters existing data will be destroyed. This is irrevocable.
* Vault does not support an automatic failover/promotion of a DR secondary cluster.
* Vault Replication is enterprise only feature.
* DR replicated cluster will replicate all data from the primary clusters, including tokens.
* A performance replicated cluster, however will not replicate the tokens from the primary

Auto-Completion Feature:

* Vault Auto-complete feature allows automatic completion for flags, sub commands and arguments.
* Be sure to restart your shell after installing autocompletion!

Installation Command:

Vault -autocomplete-install

Vault CLI commands

Enable Secrets Engine - vault secrets enable kv-v2

Store data - vault kv put secret/my-secret admin=password

list key names in secret - vault kv list secret/my-app

Delete version of secret - vault kv delete secret/my-app

Delete all versions & meta-data - vault kv metadata delete secret/my-app

Auto Unseal:

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

* The following are some of the supported services:
  - AWS KMS
  - Transit Secret Engine
  - Azure key Vault
  - HSM
  - GCP cloud kms
* For private connectivity, VPC Endpoints can be used

Identify output of Transit Engine:

Vault write encryption/encrypt/demo plaintext = $(base64 <<< "sample data")

o/p:

Vault: V3: HKVyEMMSk+29OaQ4HoOatV2rrMDJL2

* V3 indicates key version 3 was used to encrypt the plain text (3 version of keys exists)
* The name of the keyring is demo
* Transit secret engine is mounted at /encryption path

PKI Secrets Engine:

PKI secrets engine generates dynamic X.509 certificates.

Benefits of PKI Secrets Engines:

* Vault can act as a intermediate CA
* Reducing (or) eliminating certificate revocations.
* Reduces time to get a certificate by eliminating the need to generate a private key and CSR

TOTP Secrets Engine:

* TOTP stands for Time-based one-time passcodes
* These are temporary passcodes and they typically expire after 30, 60, 120 (or) 240 seconds.
* The TOTP secrets engine can act as both a generator (like Google Authenticator) and a provider (like the Google.com sign-in service)

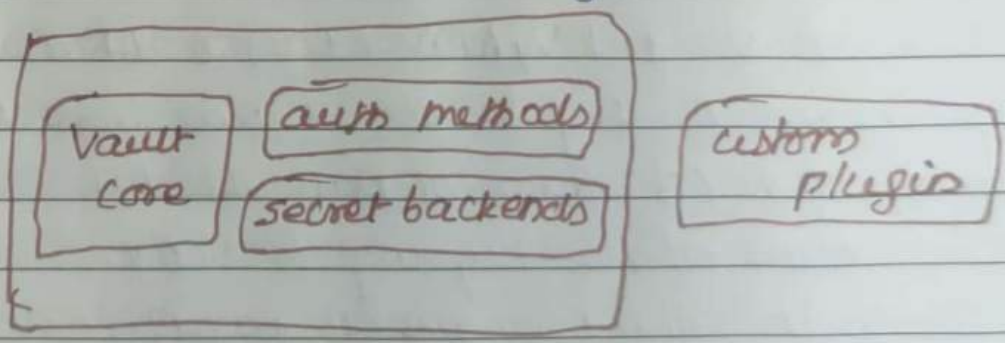
Vault read totp/code/zeal

o/p:

Code : 357893

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.



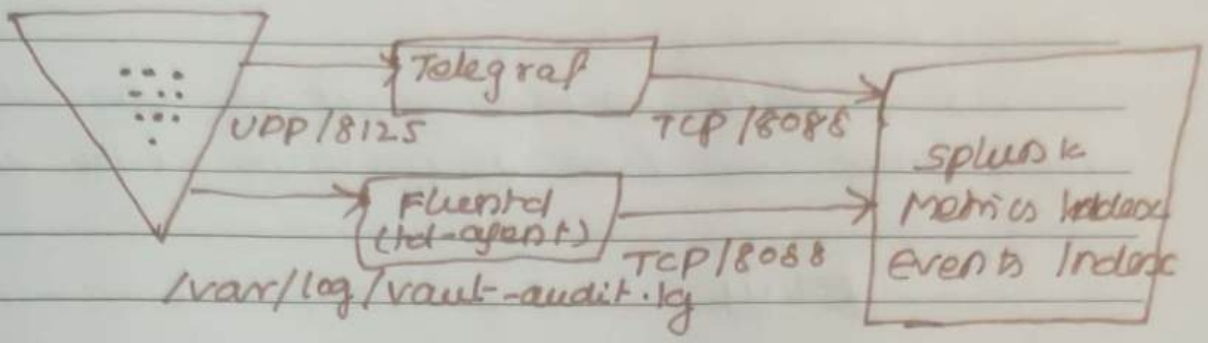
Vault

Telemetry in Vault:

Telemetry related metrics are available at /sys/metrics endpoint

Important metrics & logs to be analyzed:

* Metrics
* Vault audit logs



Security Best practices - Root tokens

* It is generally considered a best practice to not persist root tokens
* Instead, a root token should be generated using Vault's operator generate-root command only when absolutely necessary
* For day-to-day operations, the root token should be deleted after configuring other auth methods.

Enabling Versioning in KV - Version 1

* When you enable KV version 1, the versioning features not enabled by default.
* The kv enable-versioning command turns on versioning for an existing non-versioned key/value secrets engine (KV version 1) at its path
* This command also upgrades the KV from V1 to V2

Response Wrapping from UI:

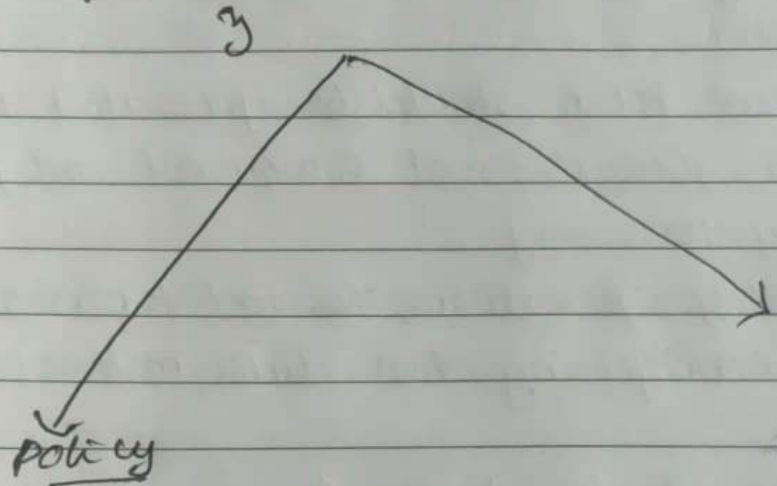
This feature under Vault tools allows you to archive response wrapping functionality

Path Templating:

Path templating allows variable replacement based on information of the entity.

policy

```
path "secret/data/{{identity.entity.name}}/*" {
    capabilities = ["create", "update"]
}
```



policy

```
path "secret/data/alice-user/*" {
capabilities = ["create", "update"]
}
```

policy

```
path "secret/data/blob-user/*
capabilities ["create", "upda
}
```

Vault tools:

Vault contains a certain set of tools that will allow us to achieve a specific function.

These are also available in the sys/tools endpoint

Endpoints	Descriptions
/sys/tools/random/164	Generate 164 bytes of random value
/sys/tools/hash/sha2-512	Hash input data based on SHA2

Service Tokens Vs Batch Tokens

	Service Tokens	Batch Tokens
Can be Root tokens	yes	No
Can creaul Child tokens	yes	No
Can Renewable	yes	No
Can be periodic	yes	No
Can have explicit Max TTL	yes	No (always uses a fixed TTL)
Has Accessors	yes	no
Has Cubbyhole	yes	No
Revoked with parent (if not orphan)	yes	Stops Working
Dynamic secrets lease Assignment	self	parent (if not orphan)
Can be used Across performance Replication Clusters	No	yes (if orphan)
creation scales with performance standby Node count	No	Yes
cost	Heavyweight, multiple storage writes per token creation	Lightweight, no storage cost for token creation

Miscellaneous points-1:

* root and default are the two default policies in vault
* When a lease is revoked, it invalidates that secret immediately and prevents any further renewals.
* To remove all secrets at a specific path:
  vault lease revoke -prefix <path>

* userpass auth method cannot read usernames and passwords from an external source.

Miscellaneous points-2

* The /sys/leader endpoint is used to check the high availability status and current leader of vault.
* vault operator init command initializes a vault server
* Vault configuration file can be used to configure various settings like Cluster name, storage backends, seal settings and so on. Other things like namespaces, auth methods are directly configured within the vault itself.
* The identity secrets engine is mounted by default in vault
* Storage backends are not trusted by vault

Miscellaneous points: 3

Important port Numbers:

8200 = Vault API and UI

8201 = Cluster to cluster Communication

* Root tokens in the vault are not associated with the TTL's and therefore they not do not expire.

Non-root tokens are associated with TTL's

* Default Max TTL of token is 32 days but it can be modified from the configuration file.

Miscellaneous points: 4

* Transit secret Engine does not store any data.

* if you receive the following error, make sure to set the VAULT_ADDR variable to HTTP

vault status

o/p: http: server gave HTTP response to HTTPS client

*After authentication the CLI and GUI automatically assume the token for subsequent requests.

* API on the other hand will require to copy the token and use it for all subsequent requests.

Miscellaneous points 5:

*Vault secret engines supports various cloud providers like AWS, Azure, Alibaba Cloud & GCP.

* Cubbyhole is a default secret engine that is enabled for all the users.

Miscellaneous points: 6

* While generating dynamic secrets, Vault returns the lease id which can in turn be used with commands such as Vault lease renew, revoke and others.
* Vault login command is used for authentication in CLI.
* After initializing, Vault provides the root token to the user, this is the only way to log login to Vault to configure additional auth methods.
* + and * are the two characters that are used for denoting wildcard paths in policies
* The vault configuration file supports either JSON(or) HCL

Miscellaneous points - 7:

* When data is decrypted via transit engine, the output is in base64 format.
* To fetch the original plain text data, you will have to decode the data with base64 -decode
* Vault UI needs to be enabled from the configuration file and not the CLI
* In order to renew a token, a user can issue a vault token renew command to extend the TTL.

Miscellaneous points: 8

* When vault is sealed, the only two options available are viewing the vault status and unsealing the vault. All other actions performed after the vault is unsealed and the user is authenticated
* If Vault CLI is able to access a specific path (/secrets/apps/creds) but the user is not able to view it via GUI then the issue is related to missing LIST permission so that the user can browse the path that leads to a specific key-value path.
* if the secret has been manually removed, if you try to do a vault lease revoke, it will result in an error stating "credential could not be found". In such a case, you need to use the -force flag.

Miscellaneous points: 9

* Storage backend and HTTP API are outside of the security barrier hence can't be protected.
* WAL stands for Write-Ahead-Logging. The changes are first recorded in the log, which must be written to stable storage before the changes are written to the datastore.

Dealing with large data sizes - Transit Engine

* When the data size is large (say 10 GB), we do not want to send it over the network to vault and get the encrypted data back. It will increase the latency and slow things down.
* Instead, you can generate a data key and encrypt it locally and use the same data key to decrypt it locally when needed.

Vault policy Rules - Transit Engine

Be aware of the important policy rules and capabilities related to transit engine

```
path "transit/encrypt/demo-key" {
    capabilities = ["update"]
}
path "transit/decrypt/demo-key" {
    capabilities = ["update"]
}
path "transit/keys" {
    capabilities = ["list"]
}
path "transit/keys/demo-key" {
    capabilities = ["read"]
}
```

Key Rotation in Vault:

* It is not recommend to encrypt all of your data with one encryption key.
* Transit engine allows customers to perform the encryption key rotations
* Vault maintains the versioned keyring and the operator can decide the minimum version allowed for decryption operations.

Min Decrypt Version:

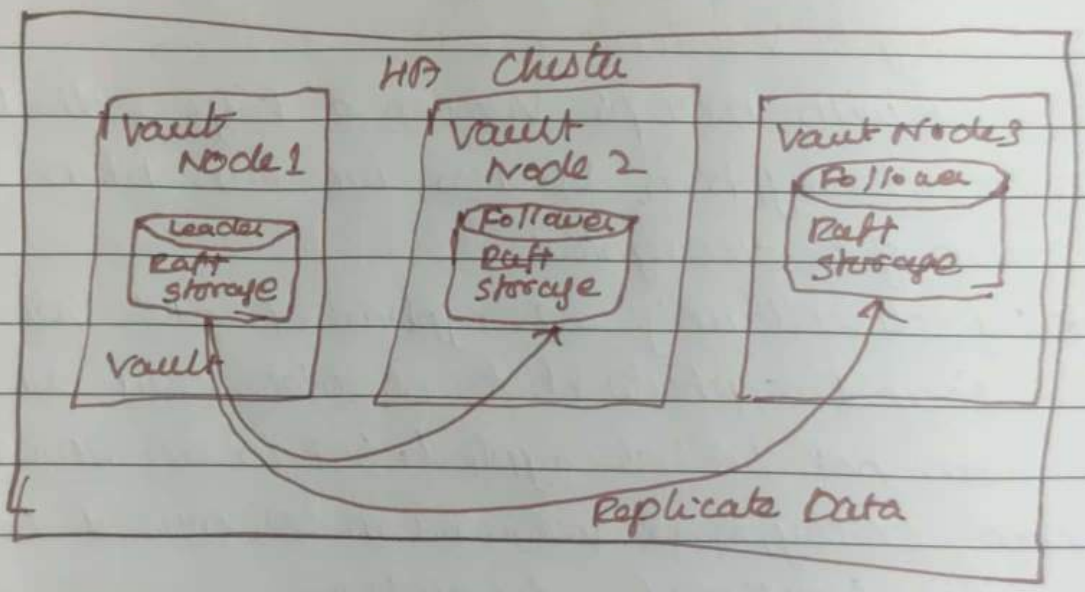
* With multiple versions of keys, with the help of min_decryption_version, we can plan on which data can get decrypted.
* By disallowing decryption of old version of keys, found ciphertext to obsolete (but sensitive) data can not be decrypted, but in an emergency, the min_decryption_version can be moved back to allow for legitimate decryption.

Periodic Tokens:

* periodic tokens never expire provided that they are renewed.
* Outside of ~~token~~ root token, it is currently the only way for a token in vault to have an unlimited lifetime.

Vault High Availability:

* Vault supports a multi-server mode for high availability. This mode protects against outages by running multiple vault servers.
* The operator step-down forces the vault server at the given address to step down from active duty.
* Consul and Integrated Storage (Raft) are some of the recommended storage backends



Guide to Vault GUI:

Vault GUI also provides the guide that gives information about various features

Vault policy format:

* The policy fmt formats a local policy file to the policy specification
* vault policy fmt file-name.hcl

Miscellaneous points:

* Kubernetes Authentication is required for kubernetes based workload.
* Note that for kubernetes Authentication, passing JWT token is not required for all the API requests
* /sys/seal endpoint is used to seal the vault. It requires a token with root policy or sudo capability on the path
* For EC2 based workloads, you can make use of AWS Authentication Method.
* Note that it is not mandatory to use the AWS auth method. You can always use other methods like AppRole.