Free Questions for HCVA0-003

Shared by Avila on 27-03-2025

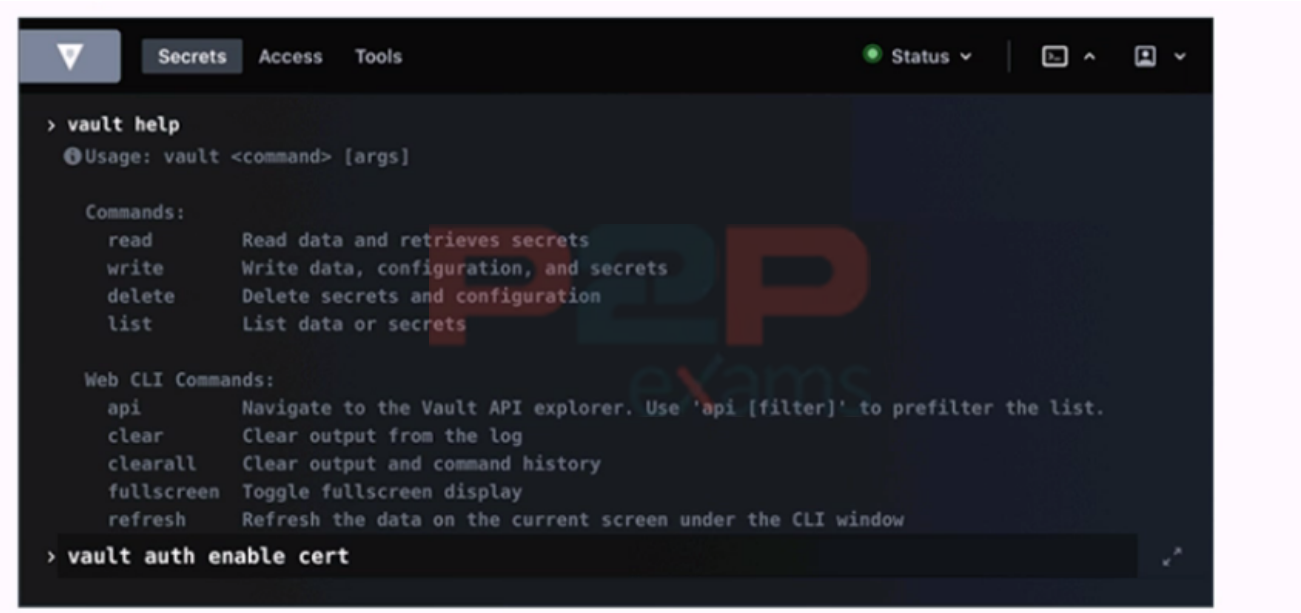
For More Free Questions and Preparation Resources

Check the Links on Last Page

Question 1

Question Type: MultipleChoice

Running the second command in the GUI CLI will succeed.



Options:

A- True

B- False

Answer:

B

Explanation:

Running the second command in the GUI CLI will fail. The second command is vault kv put secret/creds passcode=my-long-passcode. This command attempts to write a secret named creds with the value passcode=my-long-passcode to the secret path, which is the default path for the kv secrets engine. However, the kv secrets engine is not enabled at the secret path, as shown by the first command vault secrets list, which lists the enabled secrets engines and their paths. The only enabled secrets engine is the transit secrets engine at the transit path. Therefore, the second command will fail with an error message saying that no secrets engine is mounted at the path secret/. To make the second command succeed, the kv secrets engine must be enabled at the secret path or another path, using the vault secrets enable command. For example, vault secrets enable -path=secret kv would enable the kv secrets engine at the secret path.Reference:kv - Command | Vault | HashiCorp Developer,vault secrets enable - Command |

Vault | HashiCorp Developer

Question 2

Question Type: MultipleChoice

Which statement describes the results of this command: $ vault secrets enable transit

Options:

A- Enables the transit secrets engine at transit path
B- Requires a root token to execute the command successfully
C- Enables the transit secrets engine at secret path
D- Fails due to missing -path parameter
E- Fails because the transit secrets engine is enabled by default

Answer:

A

Explanation:

The command vault secrets enable transit enables the transit secrets engine at the transit path. The transit secrets engine is a secrets engine that handles cryptographic functions on data in-transit, such as encryption, decryption, signing, verification, hashing, and random bytes generation. The transit secrets engine does not store the data sent to it, but only performs the requested operations and returns the results. The transit secrets engine can also be viewed as ''cryptography as a service'' or ''encryption as a service''. The command vault secrets enable transit uses the default path of transit for the secrets engine, but this can be changed by using the -path option. For example, vault secrets enable -path=my-transit transit would enable the transit secrets engine at the my-transit path.Reference:Transit - Secrets Engines | Vault | HashiCorp Developer,vault secrets enable - Command | Vault | HashiCorp Developer

Question 3

Question Type: MultipleChoice

An organization would like to use a scheduler to track & revoke access granted to a job (by Vault)

at completion. What auth-associated Vault object should be tracked to enable this behavior?

Options:

A- Token accessor
B- Token ID
C- Lease ID
D- Authentication method

Answer:

C

Explanation:

A lease ID is a unique identifier that is assigned by Vault to every dynamic secret and service type authentication token. A lease ID contains information such as the secret path, the secret version, the secret type, etc. A lease ID can be used to track and revoke access granted to a job by Vault at completion, as it allows the scheduler to perform the following operations:

Lookup the lease information by using the vault lease lookup command or the sys/leases/lookup API endpoint. This will return the metadata of the lease, such as the expire time, the issue time, the renewable status, and the TTL.

Renew the lease if needed by using the vault lease renew command or the sys/leases/renew API endpoint. This will extend the validity of the secret or the token for a specified increment, or reset the TTL to the original value if no increment is given.

Revoke the lease when the job is completed by using the vault lease revoke command or the sys/leases/revoke API endpoint. This will invalidate the secret or the token immediately and prevent any further renewals. For example, with the AWS secrets engine, the access keys will be deleted from AWS the moment a lease is revoked.

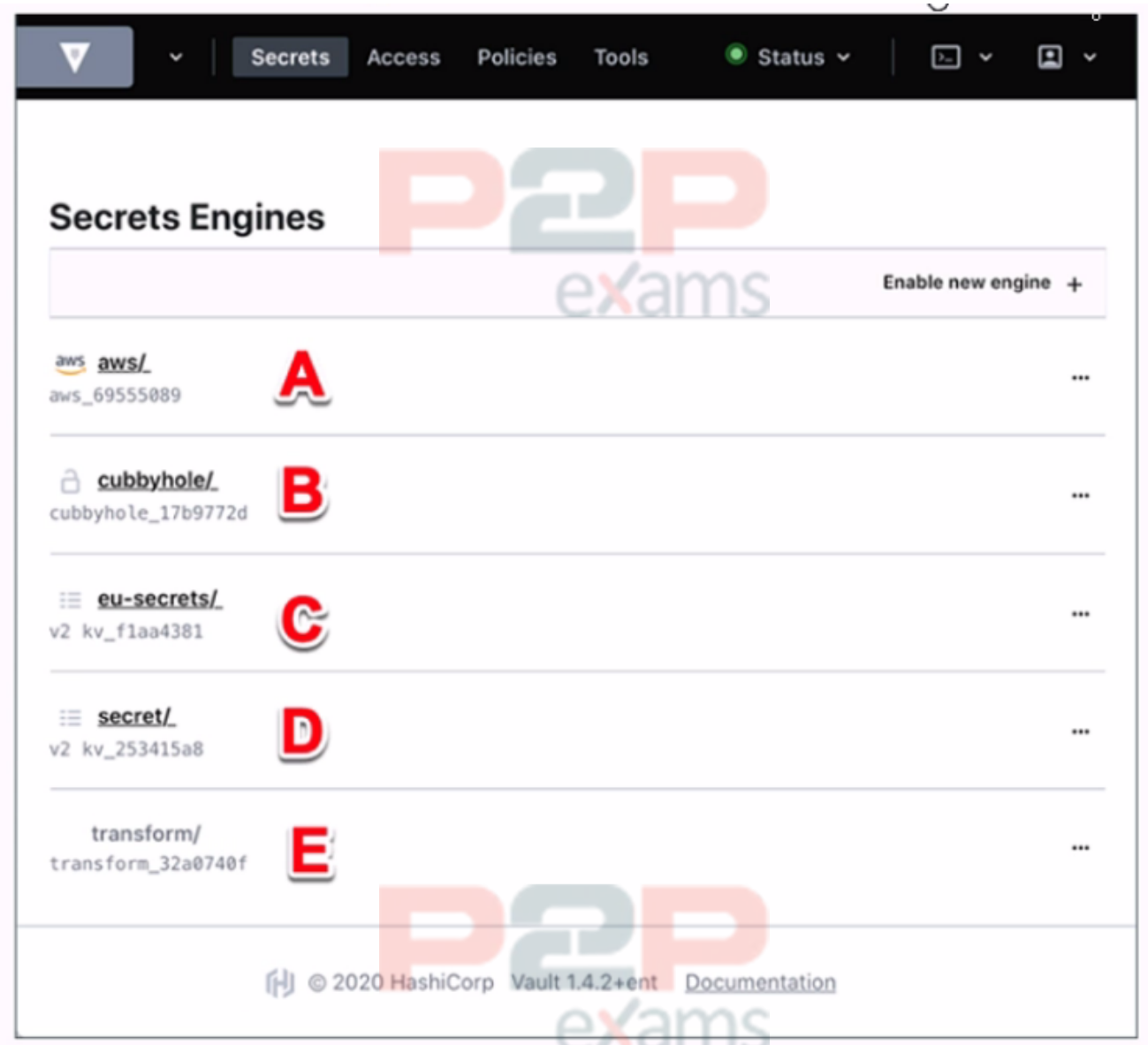
A lease ID is different from a token ID or a token accessor. A token ID is the actual value of the token that is used to authenticate to Vault and perform requests. A token ID should be treated as a secret and protected from unauthorized access. A token accessor is a secondary identifier of the token that is used for token management without revealing the token ID. A token accessor can be used to lookup, renew, or revoke a token, but not to authenticate to Vault or access secrets. A token ID or a token accessor can be used to revoke the token itself, but not the leases associated with the token. To revoke the leases, a lease ID is required.

An authentication method is a way to verify the identity of a user or a machine and issue a token with appropriate policies and metadata. An authentication method is not an object that can be tracked or revoked, but a configuration that can be enabled, disabled, tuned, or customized by using the vault auth commands or the sys/auth API endpoints.

Question 4

Question Type: MultipleChoice

Use this screenshot to answer the question below:



Where on this page would you click to view a secret located at secret/my-secret?

Options:

A- A
B- B
C- C
D- D
E- E

Answer:

C

Explanation:

In the HashiCorp Vault UI, secrets are organized in a tree-like structure. To view a secret located at secret/my-secret, you would click on the ''secret/'' folder in the tree, then click on the ''my-secret'' file. In this screenshot, the ''secret/'' folder is located at option C. This folder contains the secrets that are stored in the key/value secrets engine, which is the default secrets engine in Vault. The key/value secrets engine allows you to store arbitrary secrets as key/value pairs. The key is the path of the secret, and the value is the data of the secret. For example, the secret located at secret/my-secret has a key of ''my-secret'' and a value of whatever data you stored there.

[KV - Secrets Engines | Vault | HashiCorp Developer]

Question 5

Question Type: MultipleChoice

Which of the following statements are true about Vault policies? Choose two correct answers.

Options:

A- The default policy can not be modified
B- You must use YAML to define policies
C- Policies provide a declarative way to grant or forbid access to certain paths and operations in Vault
D- Vault must be restarted in order for a policy change to take an effect
E- Policies deny by default (empty policy grants no permission)

Answer:

C, E

Explanation:

Vault policies are written in HCL or JSON format and are attached to tokens or roles by name.

Policies define the permissions and restrictions for accessing and performing operations on certain paths and secrets in Vault.Policies are deny by default, which means that an empty policy grants no permission in the system, and any request that is not explicitly allowed by a policy is implicitly denied1. Some of the features and benefits of Vault policies are:

Policies are path-based, which means that they match the request path to a set of rules that specify the allowed or denied capabilities, such as create, read, update, delete, list, sudo, etc2.

Policies are additive, which means that if a token or a role has multiple policies attached, the effective policy is the union of all the individual policies.The most permissive capability is granted if there is a conflict3.

Policies can use glob patterns, such as * and +, to match multiple paths or segments with a single rule.For example, path ''secret/*'' matches any path starting with secret/, and path ''secret/+/config'' matches any path with two segments after secret/ and ending with config4.

Policies can use templating to interpolate certain values into the rules, such as identity information, time, randomness, etc.For example, path ''secret/{{identity.entity.id}}/*'' matches any path starting with secret/ followed by the entity ID of the requester5.

Policies can be managed by using the vault policy commands or the sys/policy API endpoints.You can write, read, list, and delete policies by using these interfaces6.

The default policy is a built-in policy that is attached to all tokens by default and cannot be deleted. However, the default policy can be modified by using the vault policy write command or the sys/policy API endpoint.The default policy provides common permissions for tokens, such as renewing themselves, looking up their own information, creating and managing response-wrapping tokens, etc7.

You do not have to use YAML to define policies, as Vault supports both HCL and JSON formats.HCL is a human-friendly configuration language that is also JSON compatible, which means that JSON can be used as a valid input for policies as well8.

Vault does not need to be restarted in order for a policy change to take effect, as policies are stored and evaluated in memory. Any change to a policy is immediately reflected in the system, and any token or role that has that policy attached will be affected by the change.

Question 6

Question Type: MultipleChoice

Which Vault secret engine may be used to build your own internal certificate authority?

Options:

A- Transit
B- PKI
C- PostgreSQL
D- Generic

Answer:

B

Explanation:

The Vault secret engine that can be used to build your own internal certificate authority is the PKI secret engine. The PKI secret engine generates dynamic X.509 certificates on-demand, without requiring manual processes of generating private keys and CSRs, submitting to a CA, and waiting for verification and signing. The PKI secret engine can act as a root CA or an intermediate CA, and can issue certificates for various purposes, such as TLS, code signing, email encryption, etc. The PKI secret engine can also manage the certificate lifecycle, such as rotation, revocation, renewal, and CRL generation. The PKI secret engine can also integrate with external CAs, such as Venafi or Entrust, to delegate the certificate issuance and management.Reference:PKI - Secrets Engines | Vault | HashiCorp Developer,Build Your Own Certificate Authority (CA) | Vault - HashiCorp Learn

Question 7

Question Type: MultipleChoice

Which of the following describes the Vault's auth method component?

Options:

A- It verifies a client against an internal or external system, and generates a token with the appropriate policies attached
B- It verifies a client against an internal or external system, and generates a token with root policy
C- It is responsible for durable storage of client tokens
D- It dynamically generates a unique set of secrets with appropriate permissions attached

Answer:

A

Explanation:

The Vault's auth method component is the component that performs authentication and assigns identity and policies to a client. It verifies a client against an internal or external system, and generates a token with the appropriate policies attached. The token can then be used to access the secrets and resources that are authorized by the policies. Vault supports various auth methods, such as userpass, ldap, aws, kubernetes, etc., that can integrate with different identity providers and systems. The auth method component can also handle token renewal and revocation, as well as identity grouping and aliasing.Reference:Auth Methods | Vault | HashiCorp Developer,Authentication - Concepts | Vault | HashiCorp Developer

Question 8

Question Type: MultipleChoice

What is a benefit of response wrapping?

Options:

A- Log every use of a secret
B- Load balanc secret generation across a Vault cluster
C- Provide error recovery to a secret so it is not corrupted in transit
D- Ensure that only a single party can ever unwrap the token and see what's inside

Answer:

D

Explanation:

Response wrapping is a feature that allows Vault to take the response it would have sent to a client and instead insert it into the cubbyhole of a single-use token, returning that token instead. The client can then unwrap the token and retrieve the original response. Response wrapping has several benefits, such as providing cover, malfeasance detection, and lifetime limitation for the secret data. One of the benefits is to ensure that only a single party can ever unwrap the token and see what's inside, as the token can be used only once and cannot be unwrapped by anyone else, even the root user or the creator of the token.This provides a way to securely distribute secrets to the intended recipients and detect any tampering or interception along the way5.

The other options are not benefits of response wrapping:

Log every use of a secret: Response wrapping does not log every use of a secret, as the secret is not directly exposed to the client or the network.However, Vault does log the creation and deletion of the response-wrapping token, and the client can use the audit device to log the unwrapping operation6.

Load balance secret generation across a Vault cluster: Response wrapping does not load balance secret generation across a Vault cluster, as the secret is generated by the Vault server that receives the request and the response-wrapping token is bound to that server.However, Vault does support high availability and replication modes that can distribute the load and improve the performance of the cluster7.

Provide error recovery to a secret so it is not corrupted in transit: Response wrapping does not provide error recovery to a secret so it is not corrupted in transit, as the secret is encrypted and stored in the cubbyhole of the token and cannot be modified or corrupted by anyone. However, if the token is lost or expired, the secret cannot be recovered either, so the client should have a backup or retry mechanism to handle such cases.

Question 9

Question Type: MultipleChoice

Which of the following statements describe the secrets engine in Vault? Choose three correct answers.

Options:

A- Some secrets engines simply store and read data
B- Once enabled, you cannot disable the secrets engine
C- You can build your own custom secrets engine
D- Each secrets engine is isolated to its path
E- A secrets engine cannot be enabled at multiple paths

Answer:

A, C, D

Explanation:

Secrets engines are components that store, generate, or encrypt data in Vault. They are enabled at a specific path in Vault and have their own API and configuration. Some of the statements that

describe the secrets engines in Vault are:

Some secrets engines simply store and read data, such as the key/value secrets engine, which acts like an encrypted Redis or Memcached.Other secrets engines perform more complex operations, such as generating dynamic credentials, encrypting data, issuing certificates, etc1.

You can build your own custom secrets engine by using the plugin system, which allows you to write and run your own secrets engine as a separate process that communicates with Vault over gRPC.You can also use the SDK to create your own secrets engine in Go and compile it into Vault2.

Each secrets engine is isolated to its path, which means that the secrets engine cannot access or interact with other secrets engines or data outside its path. The path where the secrets engine is enabled can be customized and can have multiple segments.For example, you can enable the AWS secrets engine at aws/ or aws/prod/ or aws/dev/3.

The statements that are not true about the secrets engines in Vault are:

You can disable an existing secrets engine by using the vault secrets disable command or the sys/mounts API endpoint.When a secrets engine is disabled, all of its secrets are revoked and all of its data is deleted from the storage backend4.

A secrets engine can be enabled at multiple paths, with a few exceptions, such as the system and identity secrets engines. Each secrets engine enabled at a different path is independent and isolated from others.For example, you can enable the KV secrets engine at kv/ and secret/ and they will not share any data3.

Question 10

Question Type: MultipleChoice

The Vault encryption key is stored in Vault's backend storage.

Options:

A- True
B- False

Answer:

B

Explanation:

The statement is false. The Vault encryption key is not stored in Vault's backend storage, but rather in Vault's memory. The Vault encryption key is the key that is used to encrypt and decrypt the data that is stored in Vault's backend storage, such as secrets, tokens, policies, etc. The Vault encryption key is derived from the master key, which is generated when Vault is initialized. The master key is split into unseal keys using Shamir's secret sharing algorithm, and the unseal keys are distributed to trusted operators. To start Vault, a quorum of unseal keys is required to reconstruct the master key and derive the encryption key. The encryption key is then kept in memory and used to protect the data in Vault's backend storage. The encryption key is never written to disk or exposed via the API.Reference:Seal/Unseal | Vault | HashiCorp Developer,Key Rotation | Vault | HashiCorp Developer