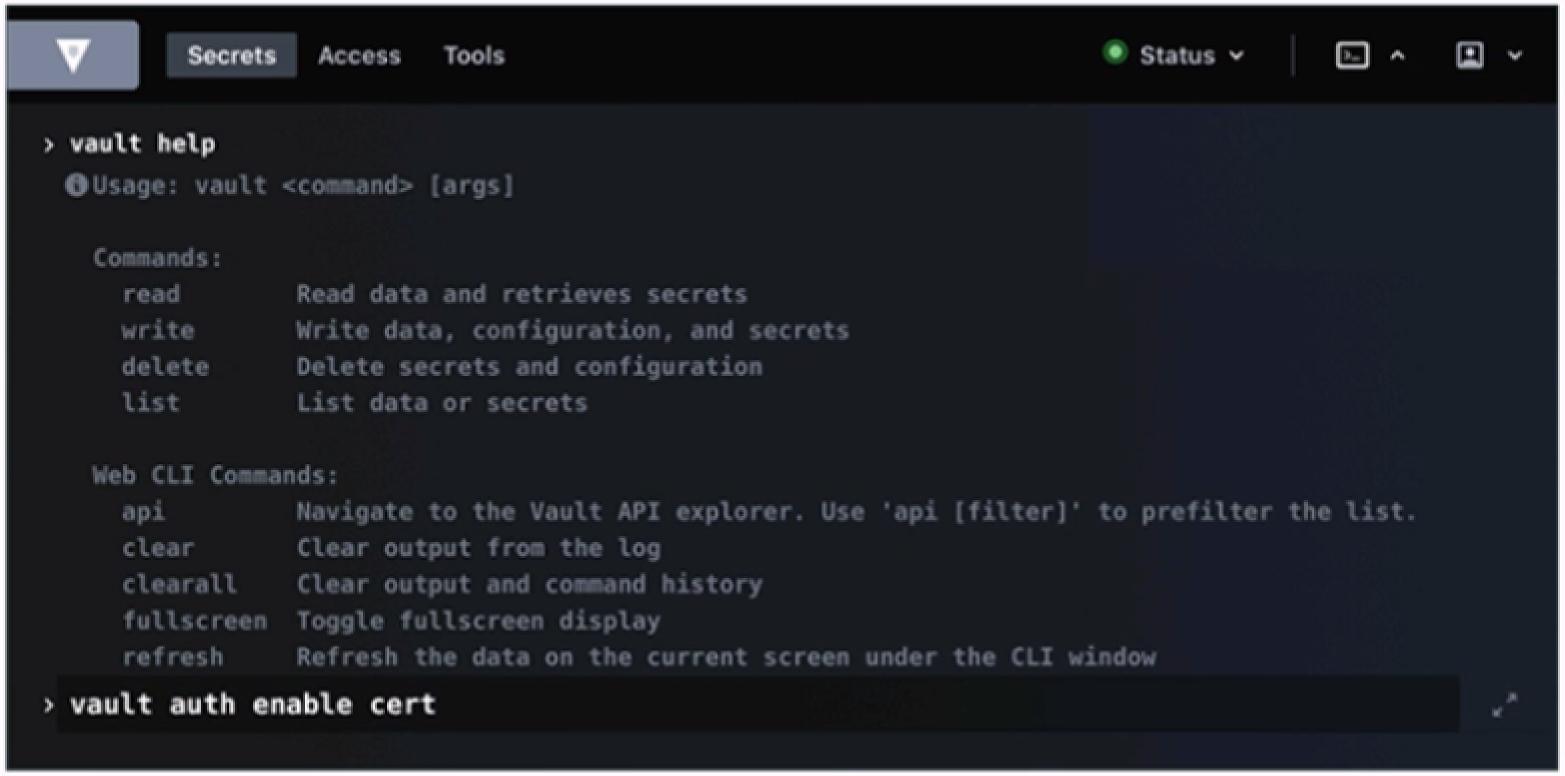
Free Questions for Vault-Associate by actualtestdumps

Shared by Clarke on 09-11-2023

Question 1

Question Type: MultipleChoice

Running the second command in the GUI CLI will succeed.



```
> vault help
  Usage: vault <command> [args]

    Commands:
      read        Read data and retrieves secrets
      write       Write data, configuration, and secrets
      delete      Delete secrets and configuration
      list        List data or secrets

    Web CLI Commands:
      api         Navigate to the Vault API explorer. Use 'api [filter]' to prefilter the list.
      clear       Clear output from the log
      clearall    Clear output and command history
      fullscreen  Toggle fullscreen display
      refresh     Refresh the data on the current screen under the CLI window
> vault auth enable cert
```

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.

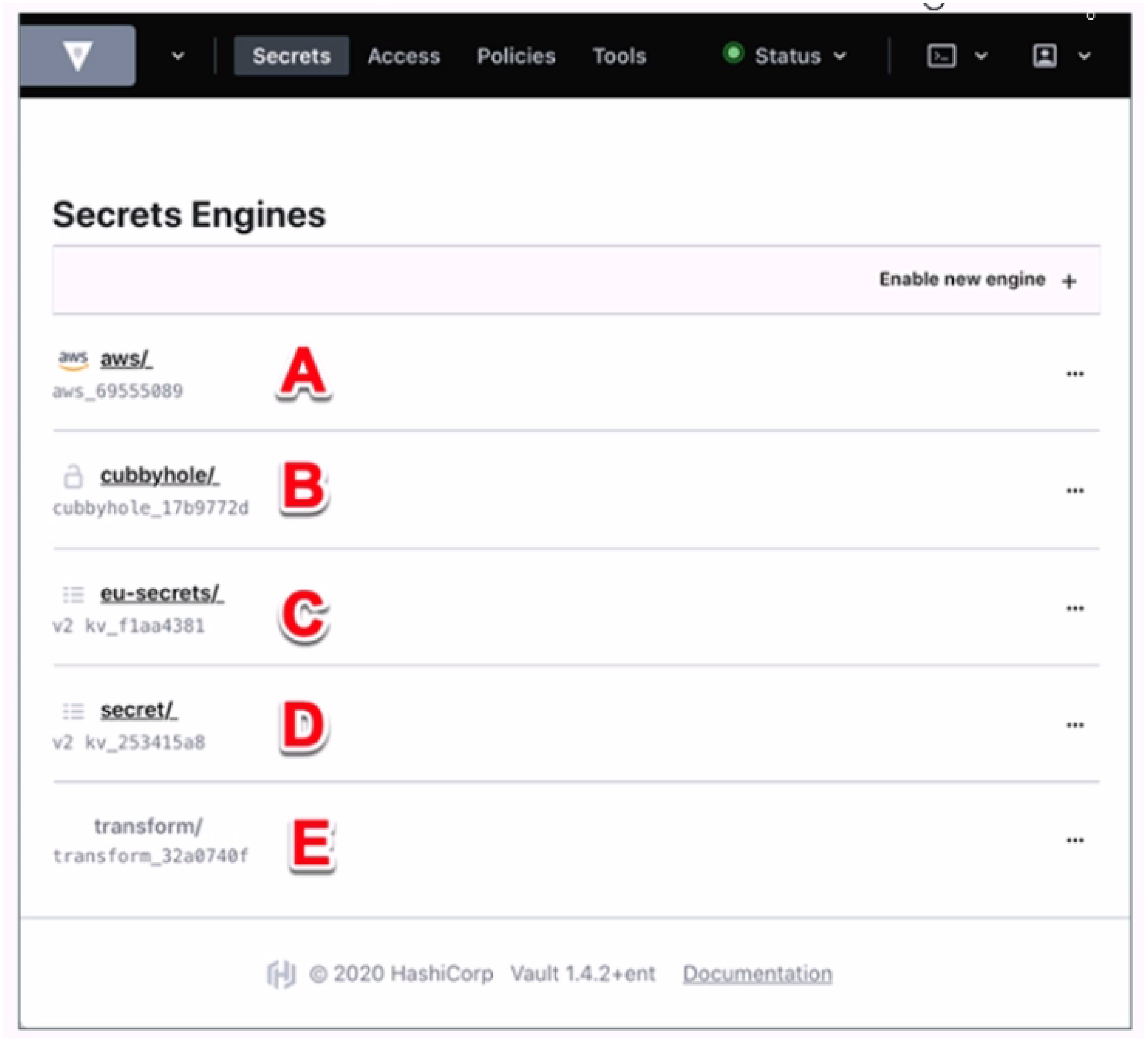
A lease ID is different from a token ID or a token accessor. A token ID is the actual value of the token that is used to authenticate to Vault and perform requests. A token ID should be treated as a secret and protected from unauthorized access. A token accessor is a secondary identifier of the token 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:



Secrets Access Policies Tools Status

Secrets Engines

Enable new engine +

aws/
aws_69555089
A

cubbyhole/
cubbyhole_17b9772d
B

eu-secrets/
v2 kv_f1aa4381
C

secret/
v2 kv_253415a8
D

transform/
transform_32a0740f
E

© 2020 HashiCorp Vault 1.4.2+ent Documentation

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**

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 7

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.

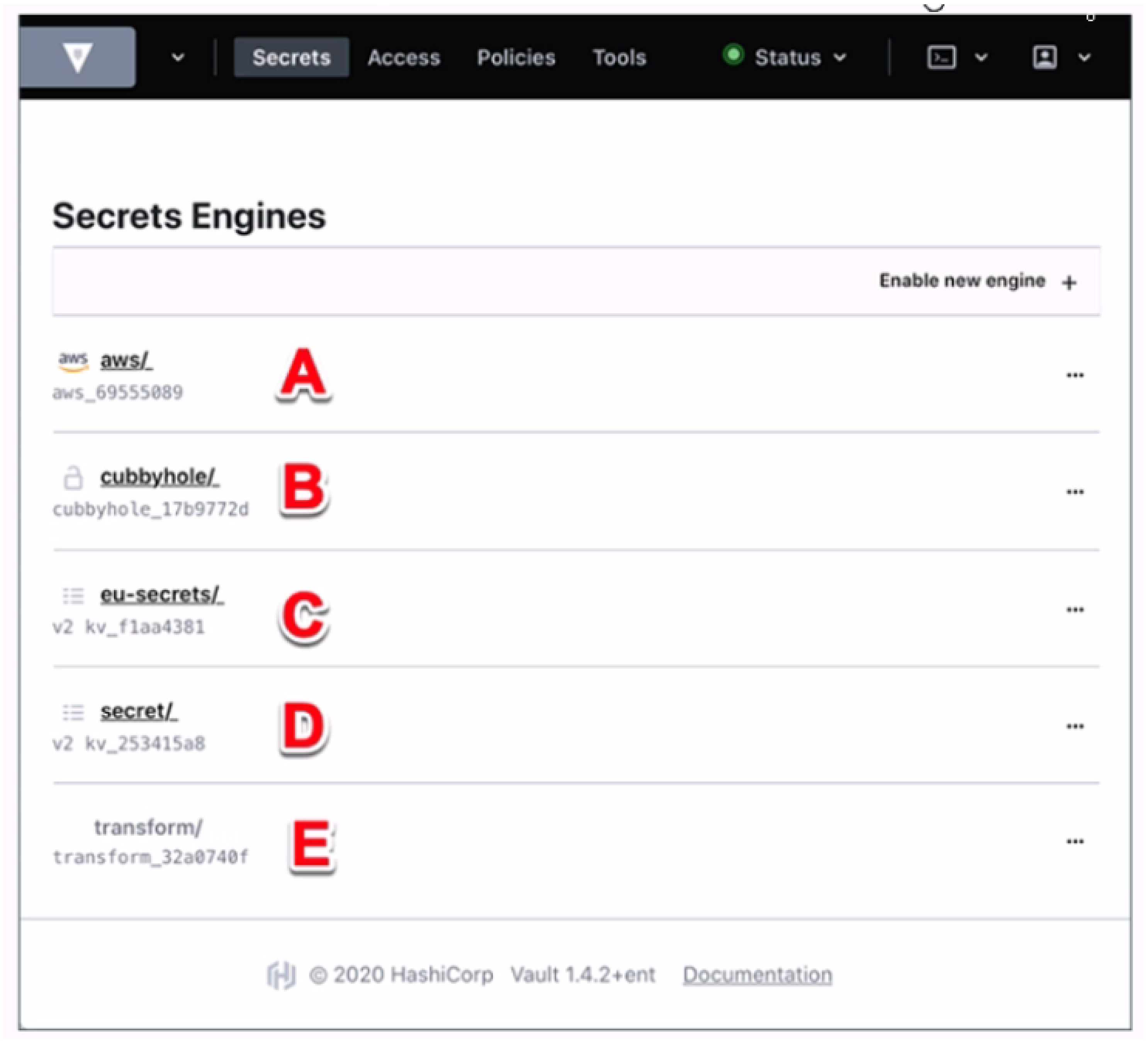
You do not have to use YAML to define policies, as Vault supports both HCL and JSON formats.HCL is a human-friendly configuration language that is also JSON compatible, which means that JSON can be used as a valid input for policies as well8.

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 8

Question Type: **MultipleChoice**

Use this screenshot to answer the question below:



Secrets Access Policies Tools Status

Secrets Engines

Enable new engine +

aws/
aws_69555089
A

cubbyhole/
cubbyhole_17b9772d
B

eu-secrets/
v2 kv_f1aa4381
C

secret/
v2 kv_253415a8
D

transform/
transform_32a0740f
E

© 2020 HashiCorp Vault 1.4.2+ent Documentation

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:

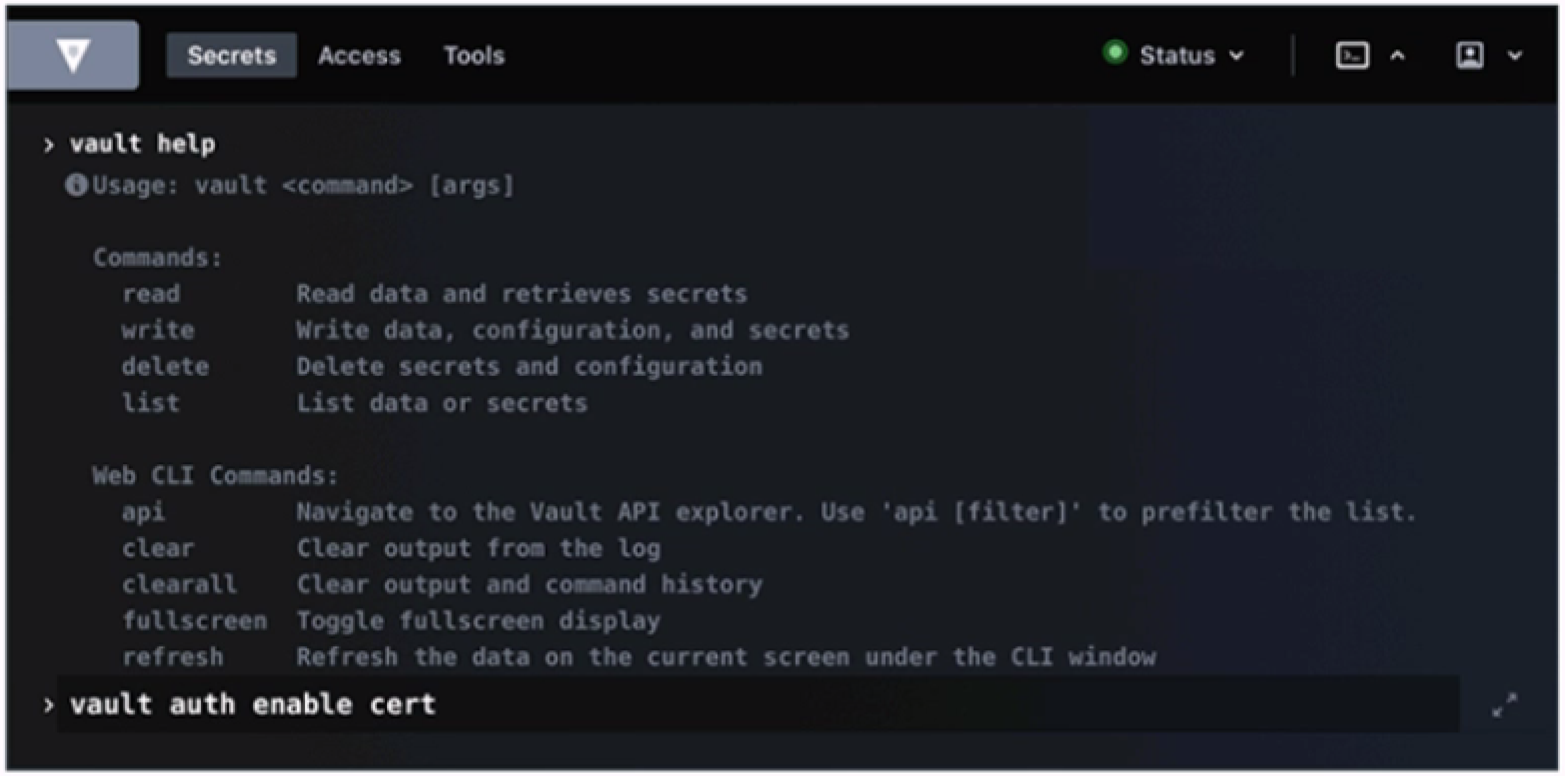
In the HashiCorp Vault UI, secrets are organized in a tree-like structure. To view a secret located at secret/my-secret, you would click on the "secret/" folder in the tree, then click on the "my-secret" file. In this screenshot, the "secret/" folder is located at option C. This folder contains the secrets 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 9

Question Type: MultipleChoice

Running the second command in the GUI CLI will succeed.



```
> vault help
  Usage: vault <command> [args]

    Commands:
      read        Read data and retrieves secrets
      write       Write data, configuration, and secrets
      delete      Delete secrets and configuration
      list        List data or secrets

    Web CLI Commands:
      api         Navigate to the Vault API explorer. Use 'api [filter]' to prefilter the list.
      clear       Clear output from the log
      clearall    Clear output and command history
      fullscreen  Toggle fullscreen display
      refresh     Refresh the data on the current screen under the CLI window
> vault auth enable cert
```

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