HashiCorp HCVA0-003 Practice Questions

Shared by Alford on 18-08-2026

For More Free Questions and Preparation Resources

Check the Links on Last Page

Question 1

Question Type: MultipleChoice

Your DevOps team would like to provision VMs in GCP via a CICD pipeline. They would like to integrate Vault to protect the credentials used by the tool. Which secrets engine would you recommend?

Options:

A- Google Cloud Secrets Engine
B- Identity secrets engine
C- Key/Value secrets engine version 2
D- SSH secrets engine

Answer:

A

Explanation:

The Google Cloud Secrets Engine is the best option for the DevOps team to provision VMs in GCP via a CICD pipeline and integrate Vault to protect the credentials used by the tool. The Google Cloud Secrets Engine can dynamically generate GCP service account keys or OAuth tokens based on IAM policies, which can be used to authenticate and authorize the CICD tool to access GCP resources. The credentials are automatically revoked when they are no longer used or when the lease expires, ensuring that the credentials are short-lived and secure. The DevOps team can configure rolesets or static accounts in Vault to define the scope and permissions of the credentials, and use the Vault API or CLI to request credentials on demand. The Google Cloud Secrets Engine also supports generating access tokens for impersonated service accounts, which can be useful for delegating access to other service accounts without storing or managing their keys1.

The Identity Secrets Engine is not a good option for this use case, because it does not generate GCP credentials, but rather generates identity tokens that can be used to access other Vault secrets engines or namespaces2. The Key/Value Secrets Engine version 2 is also not a good option, because it does not generate dynamic credentials, but rather stores and manages static secrets that the user provides3. The SSH Secrets Engine is not a good option either, because it does not generate GCP credentials, but rather generates SSH keys or OTPs that can be used to access remote hosts via SSH4.

Google Cloud - Secrets Engines | Vault | HashiCorp Developer

Identity - Secrets Engines | Vault | HashiCorp Developer

KV - Secrets Engines | Vault | HashiCorp Developer

SSH - Secrets Engines | Vault | HashiCorp Developer

Question 2

Question Type: MultipleChoice

A Fintech company is using Vault to store its static long-lived credentials so automated processes can quickly retrieve secrets. A user needs to add a new static secret for a new automated job. What CLI commands can be used to store a new static credential? (Choose two)

Options:

A- vault kv put kv/training/certification/vault @secrets.txt
B- vault kv write kv/training/certification/vault key=username value=bryan
C- vault kv create kv/training/certification/vault @secrets.txt
D- vault kv put -mount=secret creds passcode=my-long-passcode

Answer:

A, D

Explanation:

Comprehensive and Detailed In-Depth

To store static credentials in Vault's KV secrets engine via CLI, the vault kv put command is used.

A: vault kv put kv/training/certification/vault @secrets.txt writes data from a file (secrets.txt) to the path kv/training/certification/vault. The @ syntax reads key-value pairs from the file, a valid method per the KV docs.

D: vault kv put -mount=secret creds passcode=my-long-passcode specifies the mount (secret/) and stores passcode=my-long-passcode at secret/creds, a correct inline syntax.

B: vault kv write isn't a valid command; put is the correct verb. The key=value syntax is right but needs put.

C: vault kv create isn't a command; put is used to create or update secrets.

The KV CLI docs confirm vault kv put as the standard method, supporting both file input and inline key-value pairs.

KV Put Command

KV Secrets Engine Docs

Question 3

Question Type: MultipleChoice

As a best practice, the root token should be stored in Which option best ways?

Options:

A- Should be revoked and never stored after initial setup
B- Should be stored in configuration automation tooling
C- Should be stored in another password safe
D- Should be stored in Vault

Answer:

A

Explanation:

The root token is the initial token created when initializing Vault. It has unlimited privileges and can perform any operation in Vault. As a best practice, the root token should be revoked and never stored after initial setup. This is because the root token is a single point of failure and a potential security risk if it is compromised or leaked. Instead of using the root token, Vault operators should create other tokens with appropriate policies and roles that allow them to perform their tasks. If a new root token is needed in an emergency, the vault operator generate-root command can be used to create one on-the-fly with the consent of a quorum of unseal key holders. Reference: Tokens | Vault | HashiCorp Developer, Generate root tokens using unseal keys | Vault | HashiCorp Developer

Question 4

Question Type: MultipleChoice

A MySQL server has been deployed on Google Cloud Platform (GCP) to support a legacy application. You want to generate dynamic credentials against this MySQL server rather than

use static credentials. What Vault secrets engine would you use to accomplish this?

Options:

A- The GCP secrets engine
B- The Identity secrets engine
C- The database secrets engine
D- The Cubbyhole secrets engine

Answer:

C

Explanation:

Comprehensive and Detailed In-Depth

The Database secrets engine generates dynamic credentials for databases like MySQL, regardless of the platform. The Vault documentation states:

'Regardless of where a database server is deployed, you can use the database secrets engine to generate dynamic credentials against it. It doesn't matter what platform that server is deployed on, you would still use the database secrets engine. The database secrets engine generates database credentials dynamically based on configured roles.'

--- Vault Secrets: Databases

C: Correct. The database engine supports MySQL:

'Supported databases include PostgreSQL, MySQL, MongoDB, and more.'

--- Vault Secrets: Databases

A: GCP engine is for GCP services, not databases.

B: Identity engine manages identities, not credentials.

D: Cubbyhole is for temporary storage, not dynamic credentials.

Vault Secrets: Databases

Question 5

Question Type: MultipleChoice

Which statement best explains the role and usage of storage backends in HashiCorp Vault?

Options:

A- They store Vault's persistent data, affecting the scalability and performance of managing Vault.
B- They handle the encryption of all secrets so that Vault remains completely stateless.
C- They store only ephemeral tokens, ensuring no persistent data is ever saved.
D- They store only unseal keys, while all secret data remains in Vault's memory.

Answer:

A

Explanation:

Comprehensive and Detailed in Depth Explanatio n:

Storage backends in Vault are responsible for storing persistent data, impacting its operation. The HashiCorp Vault documentation states: 'The storage stanza configures the storage backend, which represents the location for the durable storage of Vault's information. Each backend has pros, cons, advantages, and trade-offs. For example, some backends support high availability while others provide a more robust backup and restoration process.' This includes secrets, policies, and audit logs, affecting scalability and performance.

The docs add: 'Vault uses a storage backend to persist its encrypted data, configuration, and metadata.' Option B is incorrect as encryption is handled by Vault, not the backend. C and D are wrong---backends store all persistent data, not just tokens or unseal keys. Thus, A is correct.

HashiCorp Vault Documentation - Storage Backends

Question 6

Question Type: MultipleChoice

Your Azure Subscription ID is stored in Vault and you need to retrieve it via Vault API for an automated job. The Subscription ID is stored at secret/cloud/azure/subscription. The secret is stored on a KV Version 2 secrets engine. What curl command below would successfully retrieve the latest version of the secret?

Options:

A- curl https://vault.krausen.com:8200/v1/secret/data/cloud/azure/subscription
B- curl --header 'X-Vault-Token: hvs.CbzCNJCVWt63jyzyaJakgDwz'
https://vault.krausen.com:8200/v1/secret/cloud/azure/subscription
C- curl --header 'X-Vault-Token: hvs.CbzCNJCVWt63jyzyaJakgDwz'
https://vault.krausen.com:8200/v1/secret/data/cloud/azure/subscription
D- curl --header 'X-Vault-Token: hvs.CbzCNJCVWt63jyzyaJakgDwz'
https://vault.krausen.com:8200/secret/data/cloud/azure/subscription/latest

Answer:

C

Explanation:

Comprehensive and Detailed In-Depth

For a KV v2 secrets engine, the API path to retrieve a secret's data is /v1/<mount>/data/. Here, the mount is secret/, and the path is cloud/azure/subscription, making the correct endpoint /v1/secret/data/cloud/azure/subscription. Authentication requires the X-Vault-Token header with a valid token. Option C matches this exactly and retrieves the latest version by default, as per KV v2 API behavior. Option A lacks the token. Option B omits the /data/ segment, invalid for KV v2. Option D adds /latest, which isn't a valid KV v2 endpoint. The KV v2 API docs confirm this structure.

KV v2 API Docs

Vault API Overview