Vault Administrative Setup

Once you have a vault instance or cluster unsealed, you can setup auditing, and administrative policies, with these instructions.

See this page for how to setup a vault instance or cluster: Hashicorp Vault Setup

Audit Logging

Create a folder for capturing audit logs on each node (leader and followers):

```
sudo mkdir -p /var/log/vault
sudo chown vault:vault /var/log/vault
sudo chmod 0750 /var/log/vault
```

Tell the leader node to store audit logs in the created folder:

```
vault audit enable file file_path=/var/log/vault/audit.log
```

Admin Policy

> WARNING: The initial root token has broad privilege, and bypasses all policy checks, being meant for bootstrapping and emergencies.
> You need to create an actual admin policy, so that proper security and auditing can occur.

Here, we will setup an admin policy, so that we can stop using the initial root token.

Doing so, gives us many benefits:

- Users created under the admin policy share the same privileges as the root user.
- We can issue short-lived admin tokens.
- We can audit all admin actions (audit logs will show policy=admin and the user, not just root).
- We can later, reduced privileges, instead of everyone having anonymous root access.

>

> NOTE: We only have to create the admin policy once.
> We can do it on the current leader.

Create a file called: admin.hcl:

```
sudo nano /etc/vault.d/admin.hcl
```

> NOTE: It doesn't matter where we generate the admin.hcl file, as the policy write command will pull it into the vault.
> But for simplicity, we will create it on the leader node, so we have easy tracking of what setup steps have been done.

Give the admin policy file, this content:

```
# --- token management (mint/lookup/renew/revoke/roles) ---
path "auth/token/create"         { capabilities = ["update"] }
path "auth/token/create/*"       { capabilities = ["update"] }
path "auth/token/lookup"         { capabilities = ["read", "update"] }
path "auth/token/lookup-self"    { capabilities = ["read"] }
path "auth/token/renew"          { capabilities = ["update"] }
path "auth/token/renew-self"     { capabilities = ["update"] }
path "auth/token/revoke"         { capabilities = ["update"] }
path "auth/token/revoke-self"    { capabilities = ["update"] }
path "auth/token/roles"          { capabilities = ["list", "read"] }
path "auth/token/roles/*"        { capabilities = ["create","read","update","delete","list"] }

# --- policies (so admins can maintain policies without root) ---
path "sys/policies/acl"          { capabilities = ["list"] }
path "sys/policies/acl/*"        { capabilities = ["create","read","update","delete","list"] }

# --- core admin knobs (optional but typical) ---
path "sys/audit"                 { capabilities = ["read"] }
path "sys/audit/*"               { capabilities = ["create","read","update","delete","sudo"] }
path "sys/auth"                  { capabilities = ["read"] }
path "sys/auth/*"                { capabilities = ["create","read","update","delete","sudo"] }
path "sys/mounts"                { capabilities = ["read"] }
path "sys/mounts/*"              { capabilities = ["create","read","update","delete","sudo"] }

# --- give admin wide access to secrets during bootstrap (tighten later) ---
path "*"                         { capabilities = ["create","read","update","delete","list","sudo"] }
```

Once created, save and close.

Here are some notes about the above policy:

- The top block allows an admin to perform all token management actions.
- The middle block allows admins to manipulate policies.
- The third block allows admins to perform other root actions.
- The bottom block (path "*") allows access to all paths in the vault (like root).
  - capabilities = create/read/update/delete/list - These allow full browsing and editing.
  - capabilities = sudo - allows doing system-level actions like:
    - enabling audit devices,
    - enabling auth methods,
    - mounting new secrets engines,
    - etc.

So, the above admin policy is just like root, but allows us to have named users as admins.

Enable the admin policy with this:

```
vault policy write admin admin.hcl
```

Once written, the policy is stored in RAFT, and replicated to all nodes.

We can confirm the policy with this:

```
vault policy read admin
```

Now, any tokens with the admin policy, will function with the above administrative privileges.

Creating Admin Tokens

Once the admin policy exists, you can create administrative access tokens.

See this page for how to create admin and user tokens: Vault Token Administration

To protect tokens in transit, see this page for Response-Wrapped Tokens: Vault Wrapping Token

---

Revision #3
Created 4 September 2025 02:43:22 by glwhite
Updated 4 September 2025 03:20:48 by glwhite