

# Project: Secure Secrets Management with HashiCorp Vault and Docker

## Objective

The main goal of this project was to learn how to securely store and manage application secrets (like database passwords and API keys) using HashiCorp Vault, avoiding the insecure practice of hardcoding them in the application code. A local Vault server was set up using Docker, and the complete workflow of creating and retrieving a secret was performed.

## Technologies Used

- HashiCorp Vault: A tool for securely storing and accessing secrets.
- Docker: A containerization platform to run Vault locally.
- PowerShell (Windows Terminal): To run Docker commands.

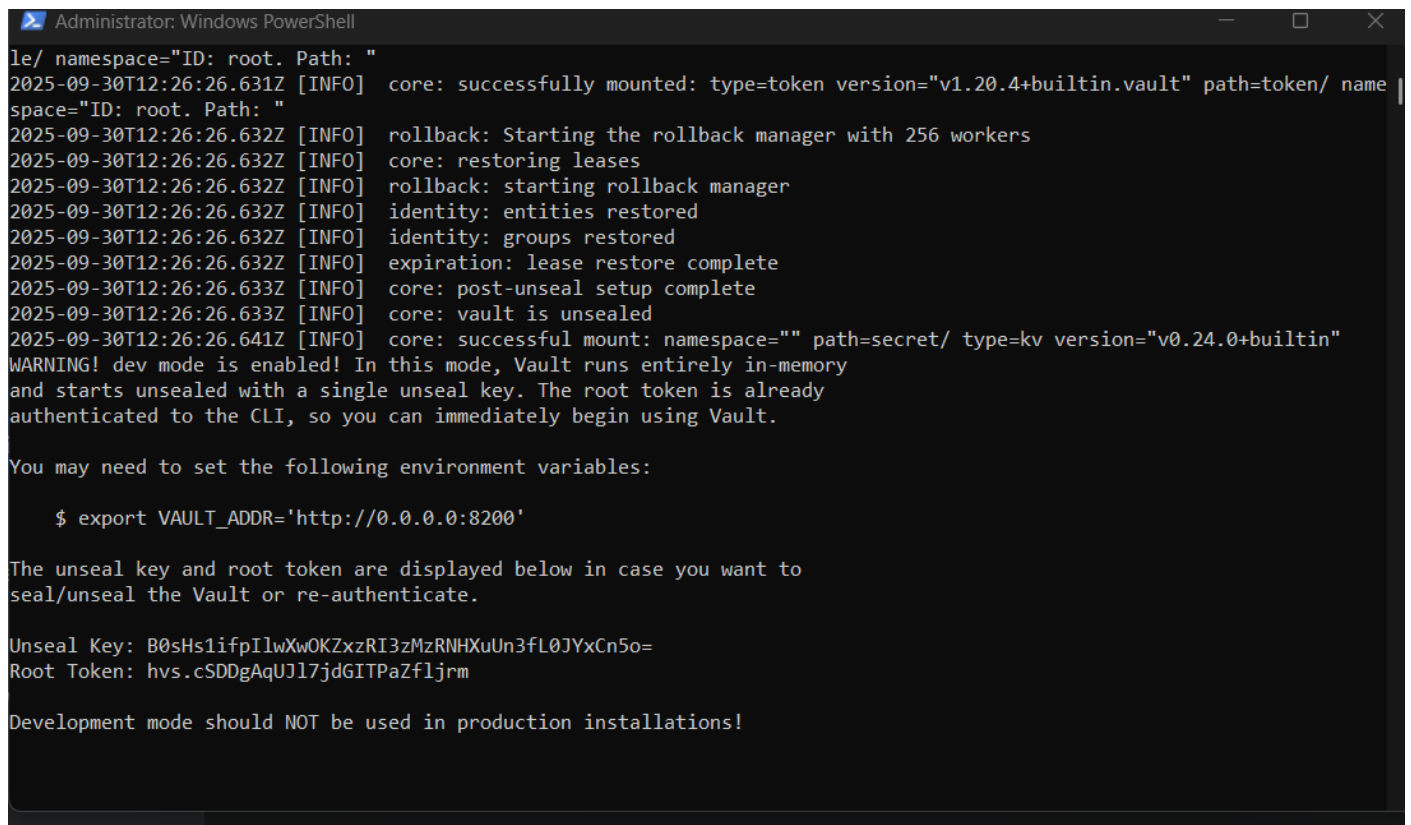
## Project Steps

### 1. Running the Vault Server

First, the Vault server was started in a Docker container using the following command:

```
docker run --cap-add=IPC_LOCK -p 8200:8200 --name dev-vault hashicorp/vault
```

The logs from this command provided the critical Unseal Key and Root Token needed to access the server.

A screenshot of a Windows PowerShell terminal window titled "Administrator: Windows PowerShell". The terminal displays the output of the Docker command to run the Vault server. The logs show the Vault server successfully mounting, restoring leases, and starting the rollback manager. It then reports that the vault is unsealed and provides the unseal key and root token. A warning message indicates that dev mode is enabled, meaning Vault runs entirely in-memory and starts unsealed with a single unseal key. The terminal also shows the environment variable VAULT\_ADDR being set to 'http://0.0.0.0:8200'.

```
le/ namespace="ID: root. Path: "
2025-09-30T12:26:26.631Z [INFO] core: successfully mounted: type=token version="v1.20.4+builtin.vault" path=token/ name
space="ID: root. Path: "
2025-09-30T12:26:26.632Z [INFO] rollback: Starting the rollback manager with 256 workers
2025-09-30T12:26:26.632Z [INFO] core: restoring leases
2025-09-30T12:26:26.632Z [INFO] rollback: starting rollback manager
2025-09-30T12:26:26.632Z [INFO] identity: entities restored
2025-09-30T12:26:26.632Z [INFO] identity: groups restored
2025-09-30T12:26:26.632Z [INFO] expiration: lease restore complete
2025-09-30T12:26:26.633Z [INFO] core: post-unseal setup complete
2025-09-30T12:26:26.633Z [INFO] core: vault is unsealed
2025-09-30T12:26:26.641Z [INFO] core: successful mount: namespace="" path=secret/ type=kv version="v0.24.0+builtin"
WARNING! dev mode is enabled! In this mode, Vault runs entirely in-memory
and starts unsealed with a single unseal key. The root token is already
authenticated to the CLI, so you can immediately begin using Vault.

You may need to set the following environment variables:

$ export VAULT_ADDR='http://0.0.0.0:8200'

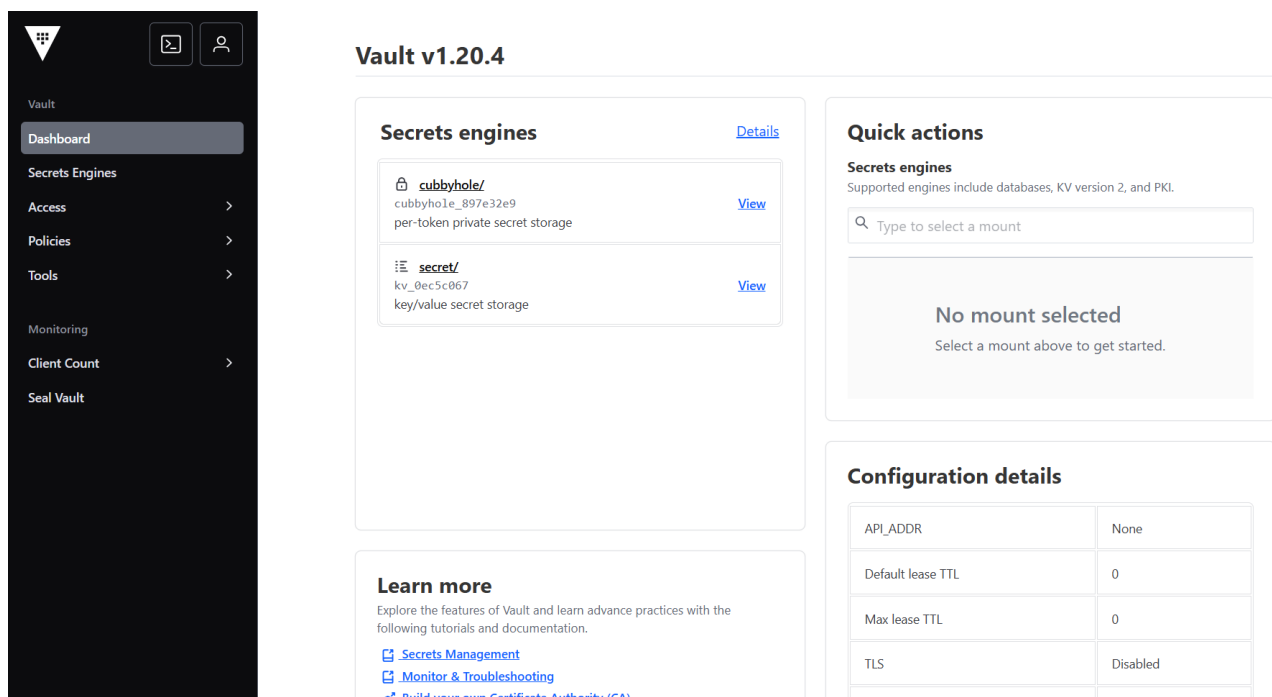
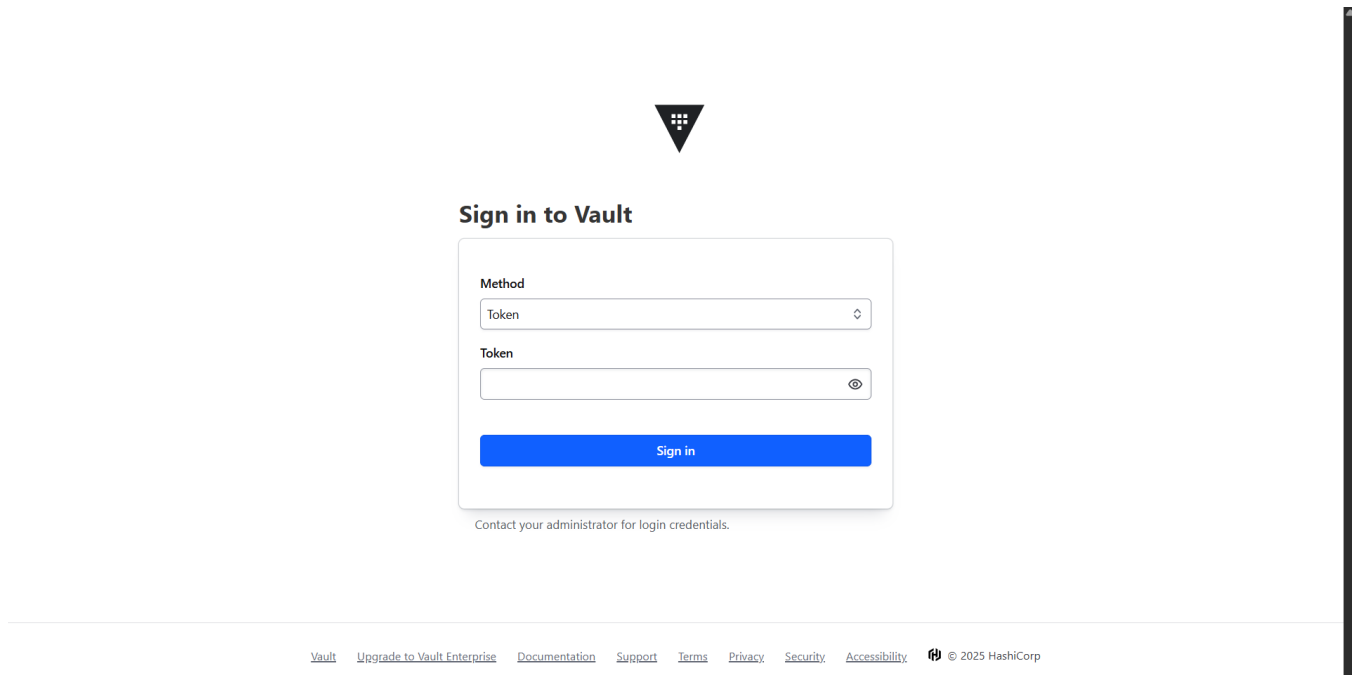
The unseal key and root token are displayed below in case you want to
seal/unseal the Vault or re-authenticate.

Unseal Key: B0sHs11fpIlwXwOKZxzRI3zMzRNHXuUn3fL0JYxCn5o=
Root Token: hvs.cSDDgAqUJl7jdGITPaZfljrm

Development mode should NOT be used in production installations!
```

## 2. UI Access and Login

With the server running, the Vault UI was accessed at <http://localhost:8200>. I logged in using the Root Token obtained from the startup logs.

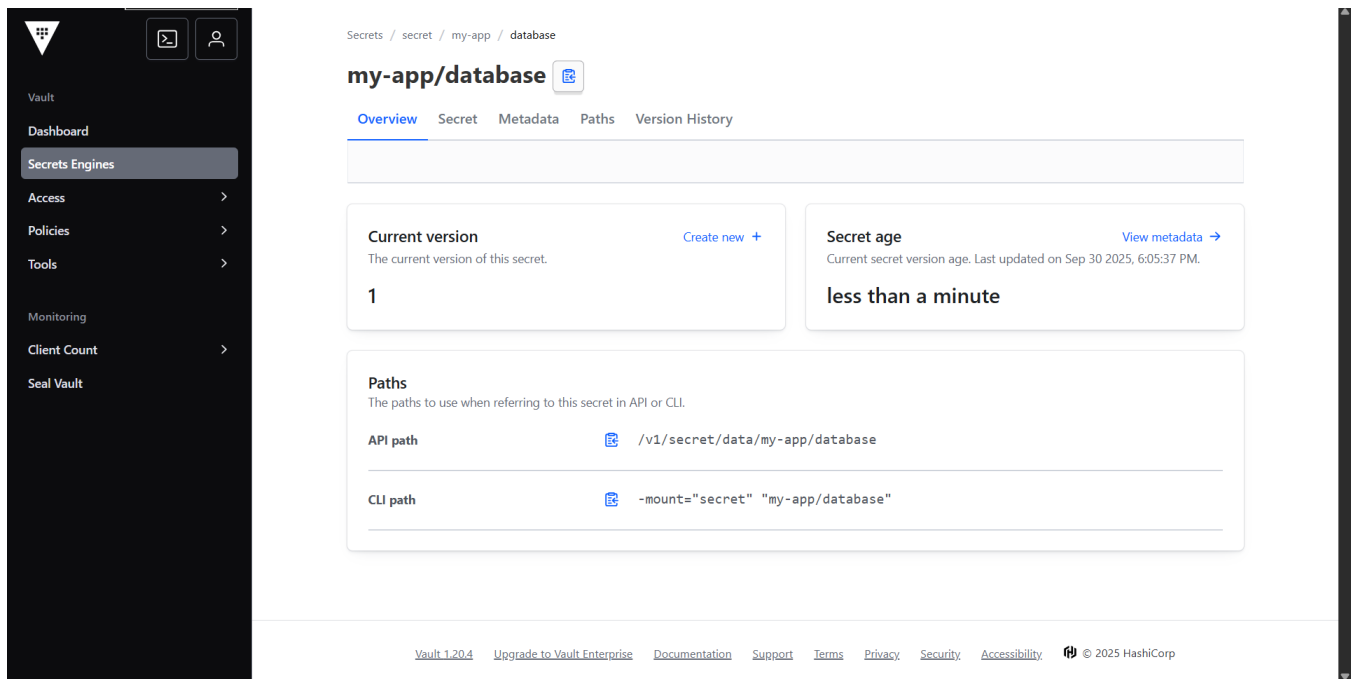
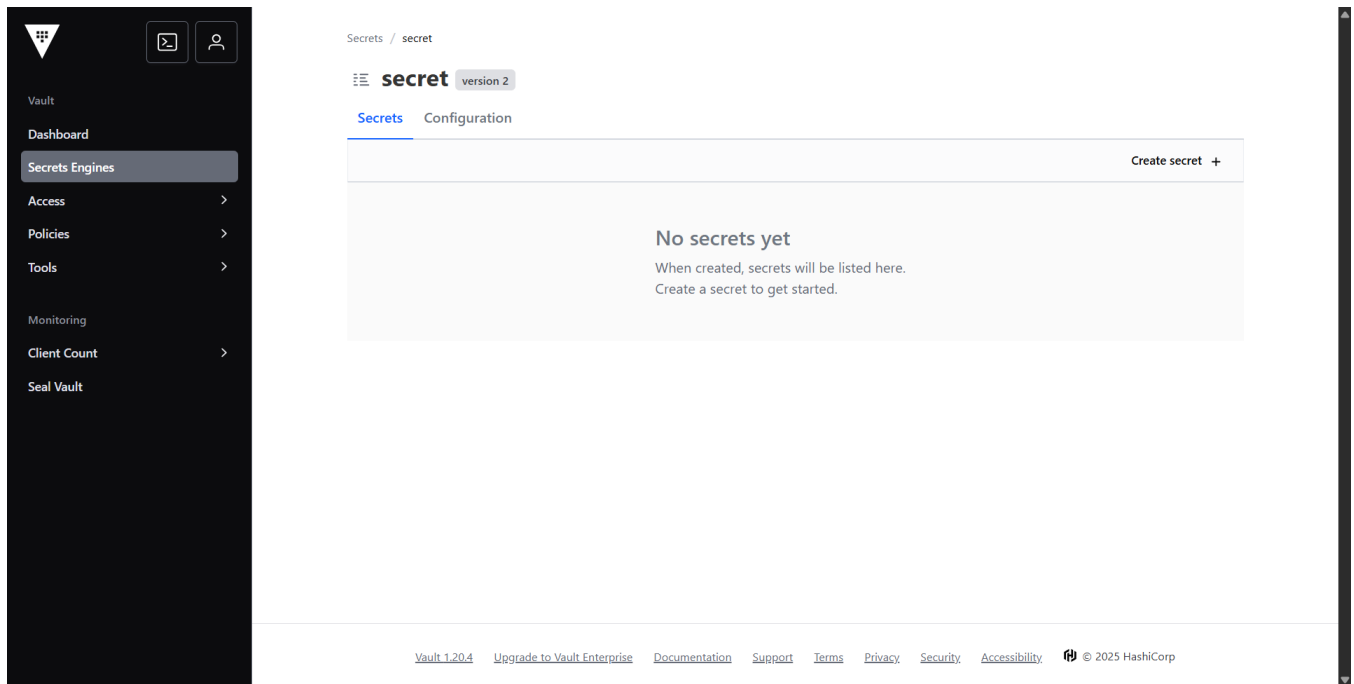


## 3. Creating a Secret

Inside the default secret/ secrets engine, a new secret was created to store sample database credentials.

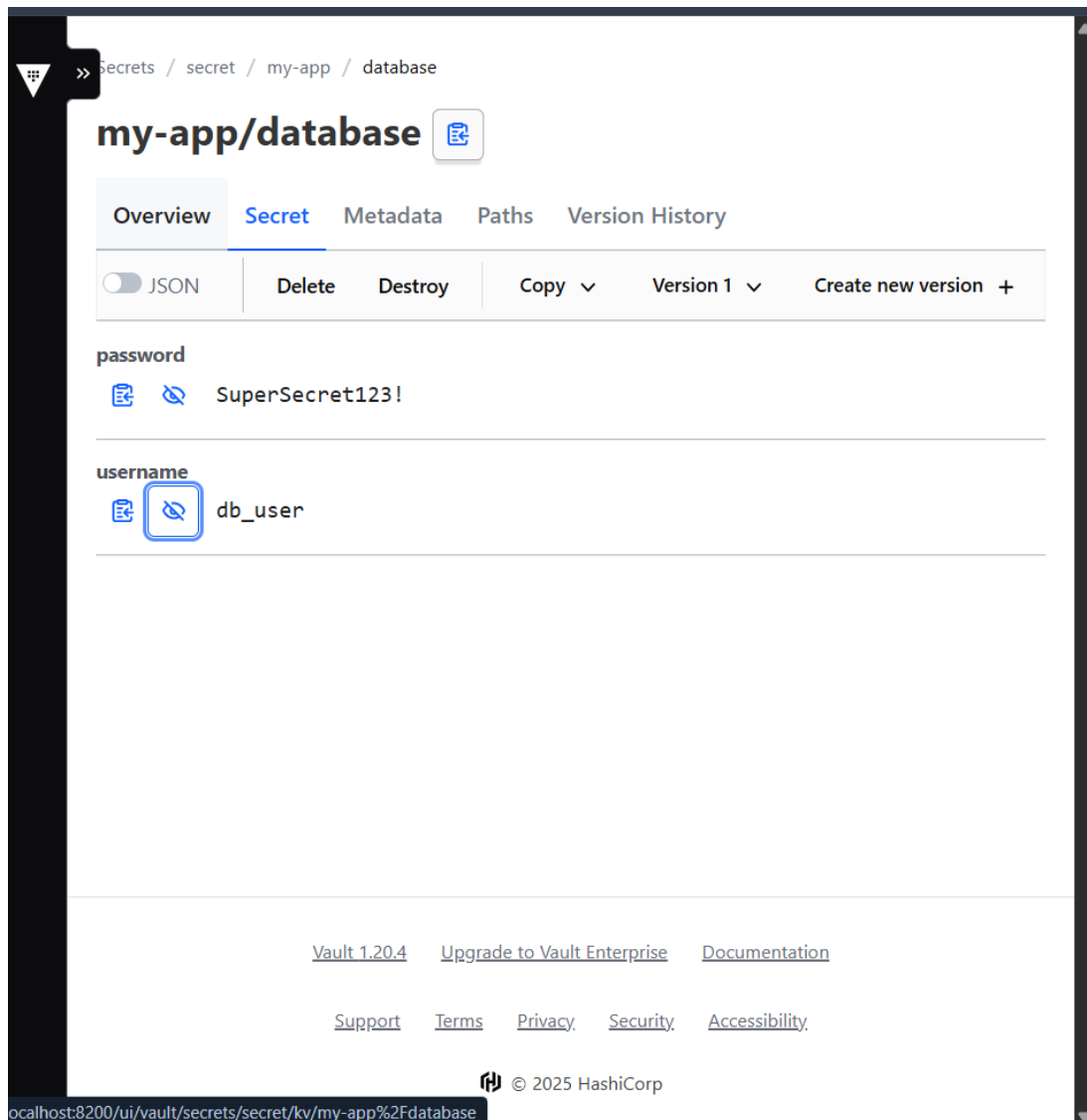
Path: my-app/database

Secret Data: - username: db\_user - password: SuperSecret123!



#### 4. Viewing the Stored Secret

The created secret and its data were viewed by navigating to the 'Secret' tab, confirming that the data was stored securely and correctly.



## Conclusion

This project provided a practical, hands-on introduction to secrets management with HashiCorp Vault. It successfully demonstrated the core workflow of running a Vault instance, authenticating, and performing basic create/read operations for secrets. This experience highlighted the importance of decoupling secrets from application code and how tools like Vault provide a secure, centralized solution for this critical task.