Vault Workshop

Modern Security with Vault for any Cloud

HashiCorp

The Slide Deck

Follow along on your own computer at this link:

https://hashicorp.github.io/field-workshops-vault/slides/multi-cloud/vault-oss/index.html

Introductions

- Your Name
- Job Title
- Secrets Management Experience
- Favorite Text Editor

Table of Contents

Lab Environment Used

- This workshop uses Instruqt for hands-on labs.
- Instruqt labs are run in "tracks" that are divided into "challenges".
- This workshop uses the following tracks:
  1. **Vault Basics**
  2. **Vault Dynamic Database Credentials**
  3. **Vault Encryption as a Service**
- We'll cover chapters 1-6 and then do the first lab.
- We'll then cover chapter 7 with the second lab.
- We'll finish with chapter 8 and the third lab.

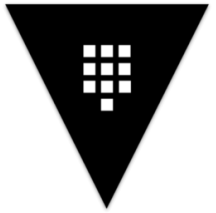
Your instructor will provide the URLs for the tracks.

Chapter 1

HashiCorp Vault Overview

HashiCorp

HashiCorp Vault Overview



- HashiCorp Vault is an API-driven, cloud agnostic secrets management system.
- It allows you to safely store and manage sensitive data in hybrid cloud environments.
- You can also use Vault to generate dynamic short-lived credentials, or encrypt application data on the fly.

The Traditional Security Model

Also known as the "Castle and Moat" method.

The Traditional Security Model

- Traditional security models were built upon the idea of perimeter based security.
- There would be a firewall, and inside that firewall it was assumed one was safe.
- Resources such as databases were mostly static. As such rules were based upon IP address, credentials were baked into source code or kept in a static file on disk.

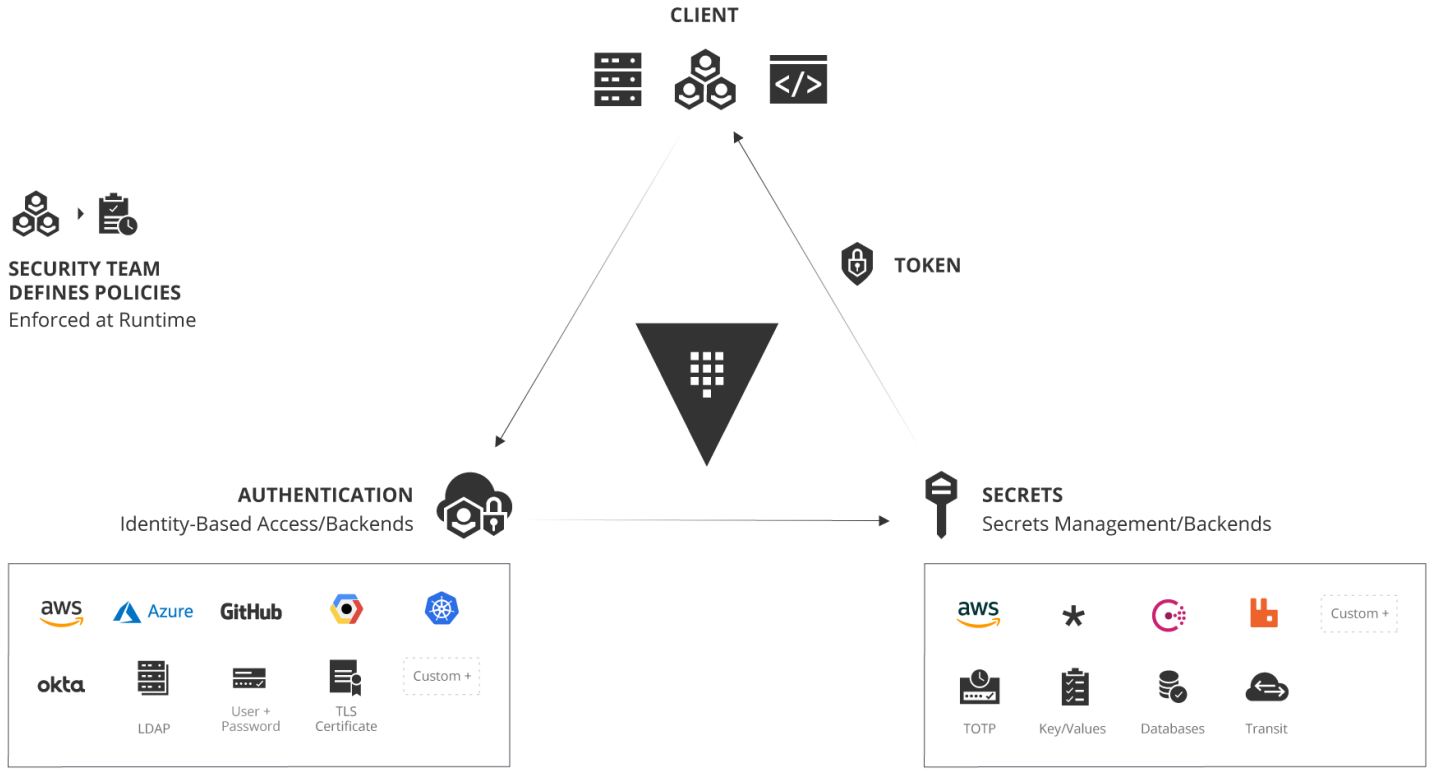
Problems with the Traditional Security Model

- IP Address based rules
- Hardcoded credentials with problems such as:
  - Shared service accounts for apps and users
  - Difficult to rotate, decommission, and determine who has access
  - Revoking compromised credentials could break

Modern Secrets Management

No well defined perimeter; security enforced by identity.

Identity Based Security



[Identity Based Security and Low Trust Networks]

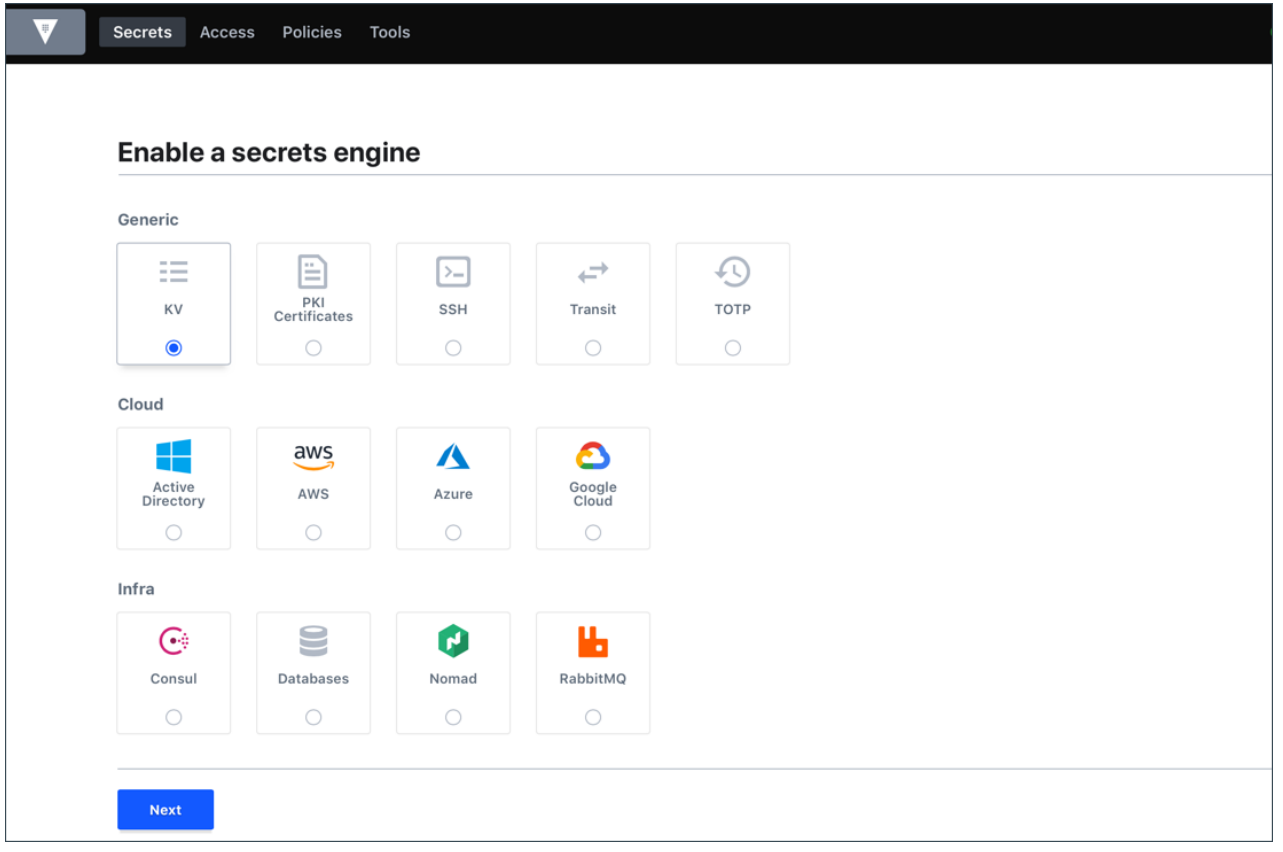
Identity Based Security

Vault was designed to address the security needs of modern applications.

It differs from the traditional approach by using:

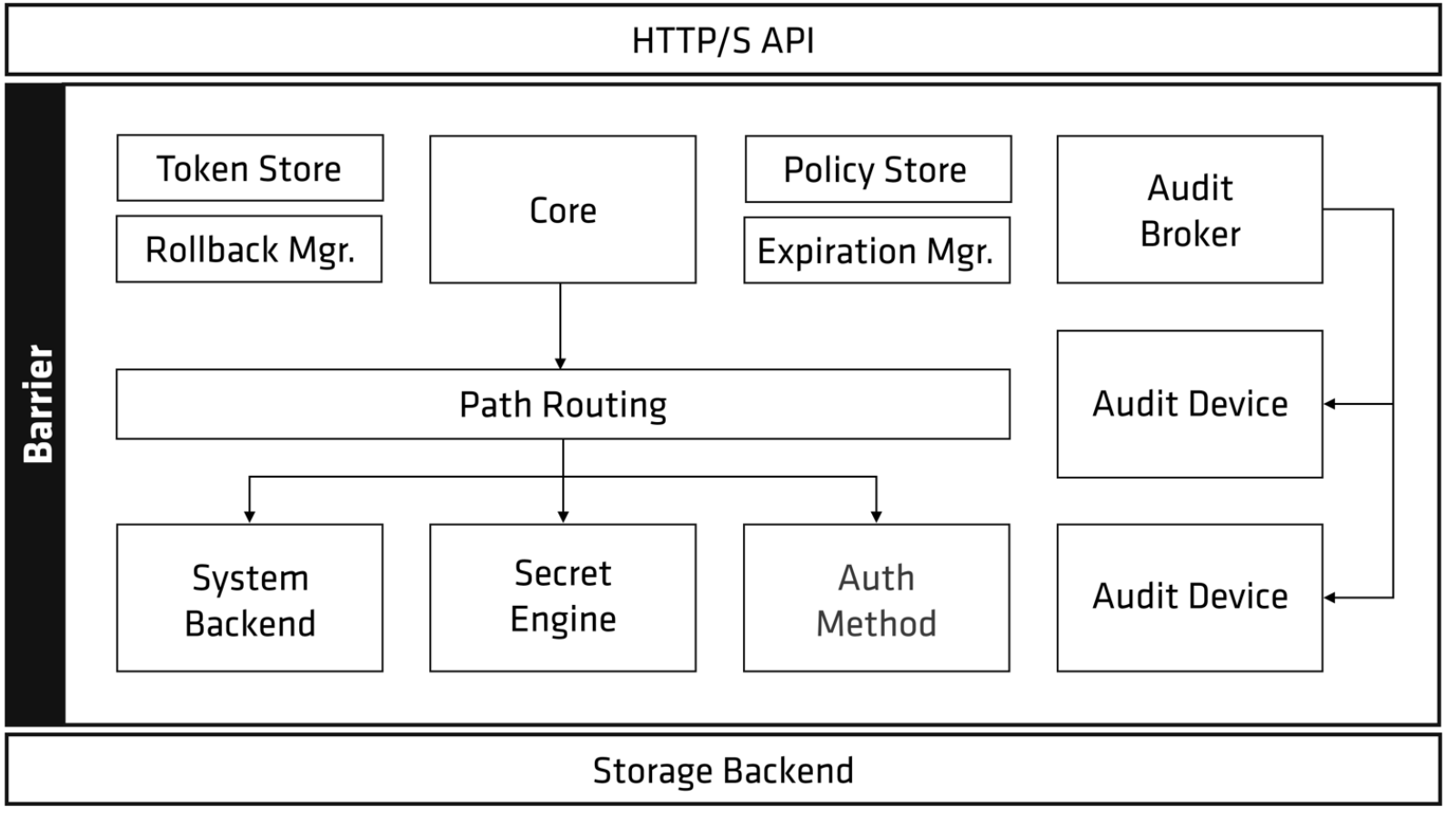
- Identity based rules allowing security to stretch across network perimeters
- Dynamic, short lived credentials that are rotated frequently
- Individual accounts to maintain provenance (tie action back to entity)
- Credentials and Entities that can easily be invalidated

Vault Secrets Engines



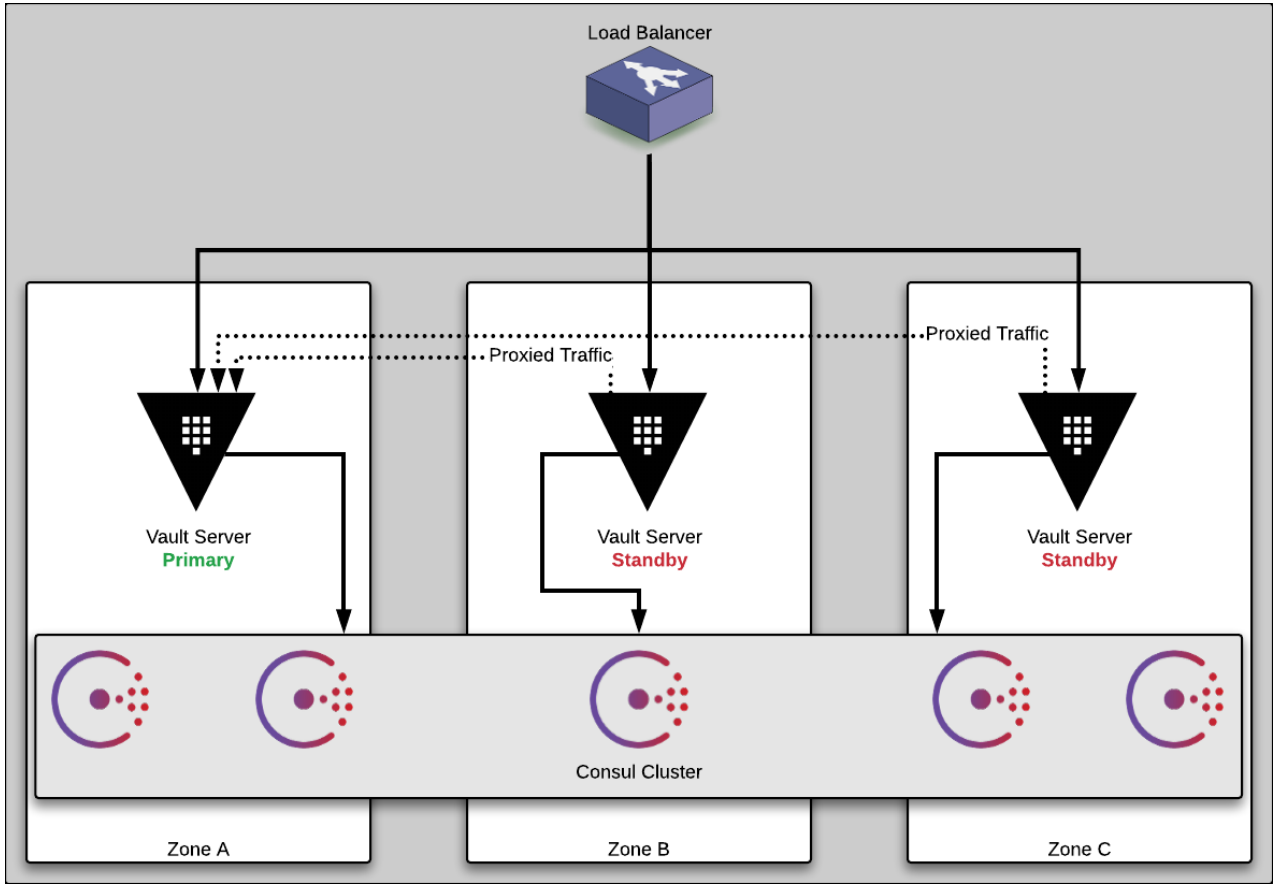
[Vault Secrets Engines]

Vault Architecture Internals



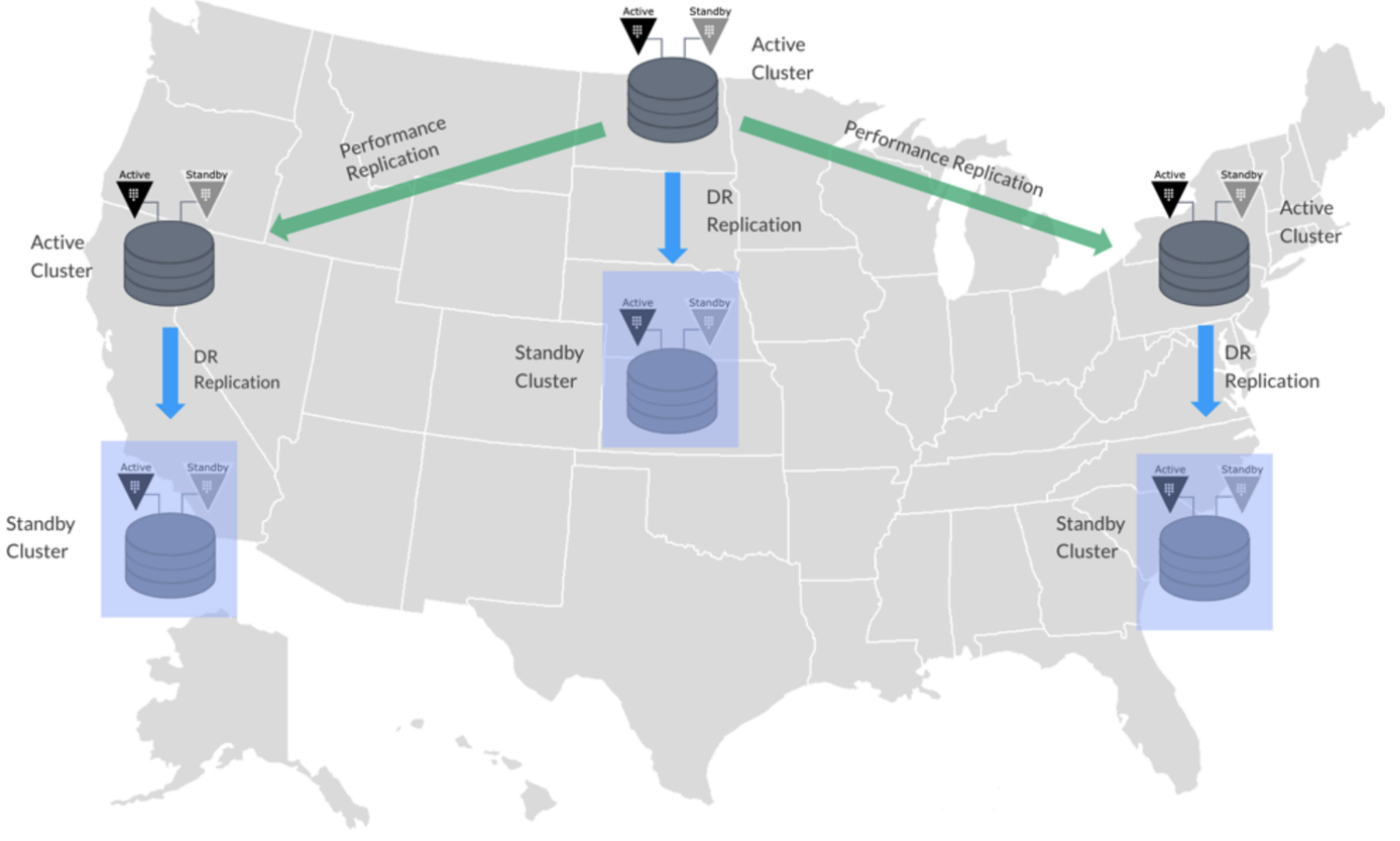
[HashiCorp Vault Internals Architecture]

Vault Architecture - High Availability



[Vault High Availability]

Vault Architecture - Multi-Region



[Vault Enterprise Replication]

📝 Chapter 1 Review

- What is HashiCorp Vault?

📝 Chapter 1 Review

- What is HashiCorp Vault?
  - Vault is a Secrets Management System.
  - It is API-driven and cloud agnostic.
  - It can be used in untrusted networks.
  - It can authenticate users and applications against many systems.
  - It supports dynamic generation of short-lived secrets.
  - It runs in highly available clusters that can be replicated across regions.

Chapter 2

Interacting With Vault

HashiCorp

Interacting With Vault

Vault provides several mechanisms for interacting with it:

- The Vault CLI
- The Vault UI
- The Vault API

The Vault CLI

- The Vault CLI is a Go application.
- It runs on macOS, Windows, Linux, and other operating systems.
- You can download the latest version here.

Installing the Vault CLI

- Installing Vault on your laptop is easy:
  - Simply download the zip file.
  - Unpack the `vault` binary.
  - Place the binary in your path.

See this tutorial for more details.

Some Basic Vault CLI Commands

- `vault` by itself will give you a list of many Vault CLI commands.
  - The list starts with the most common ones.
- `vault version` tells you the version of Vault you are running.
- `vault read` is used to read secrets from Vault.
- `vault write` is used to write secrets to Vault.

The `-h`, `-help`, and `--help` flags can be added to get help for any Vault CLI command.

Vault Server Modes

Vault servers can be run in two different modes:

- "Dev" mode that is only intended for development
- "Prod" mode that can be used in QA and production

Vault's "Dev" Mode

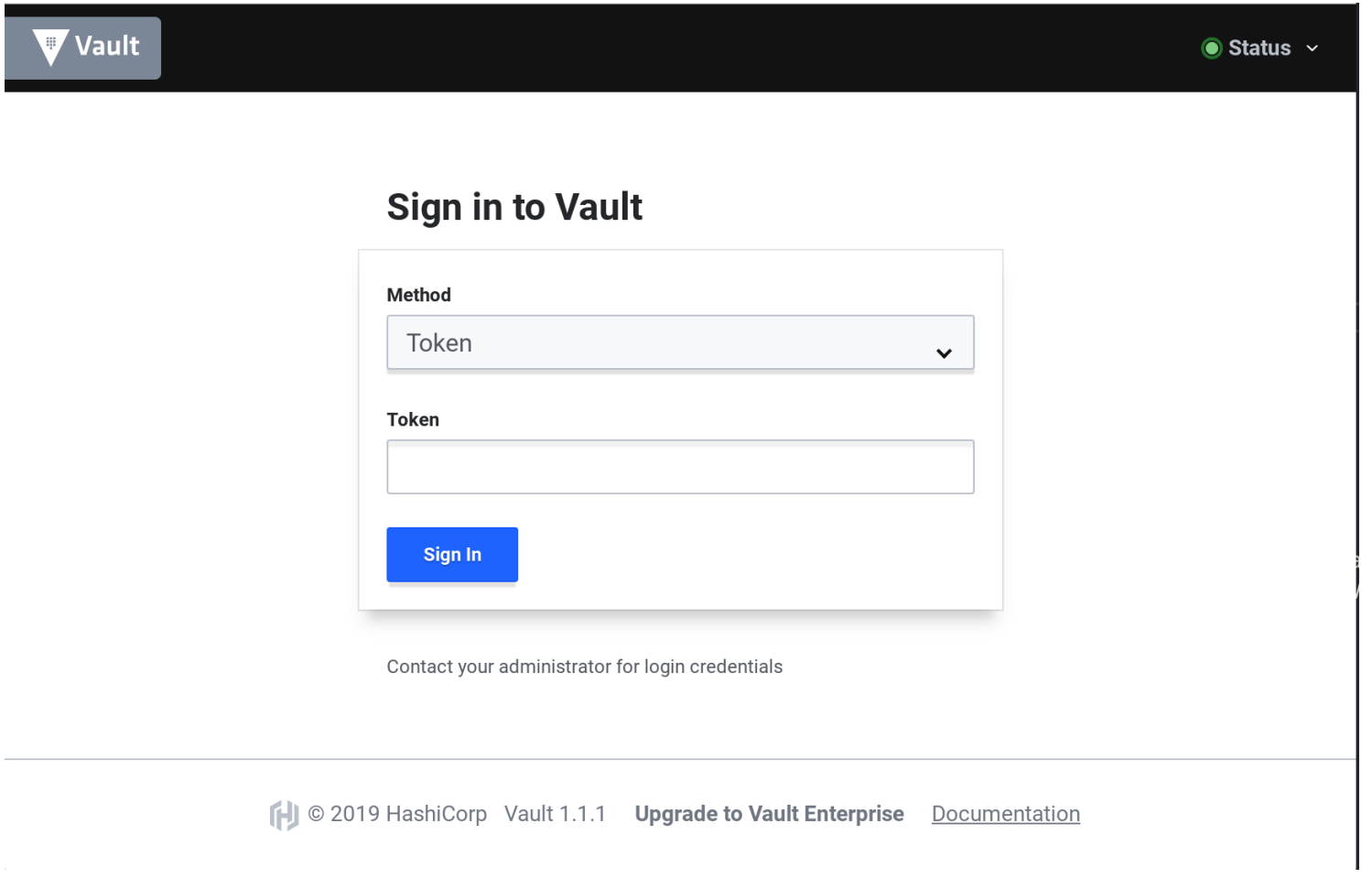
- It is not secure.
- It stores everything in memory.
- Vault is automatically unsealed.
- The root token can be specified before launching.

Please never store actual secrets on a server run in "Dev" mode.

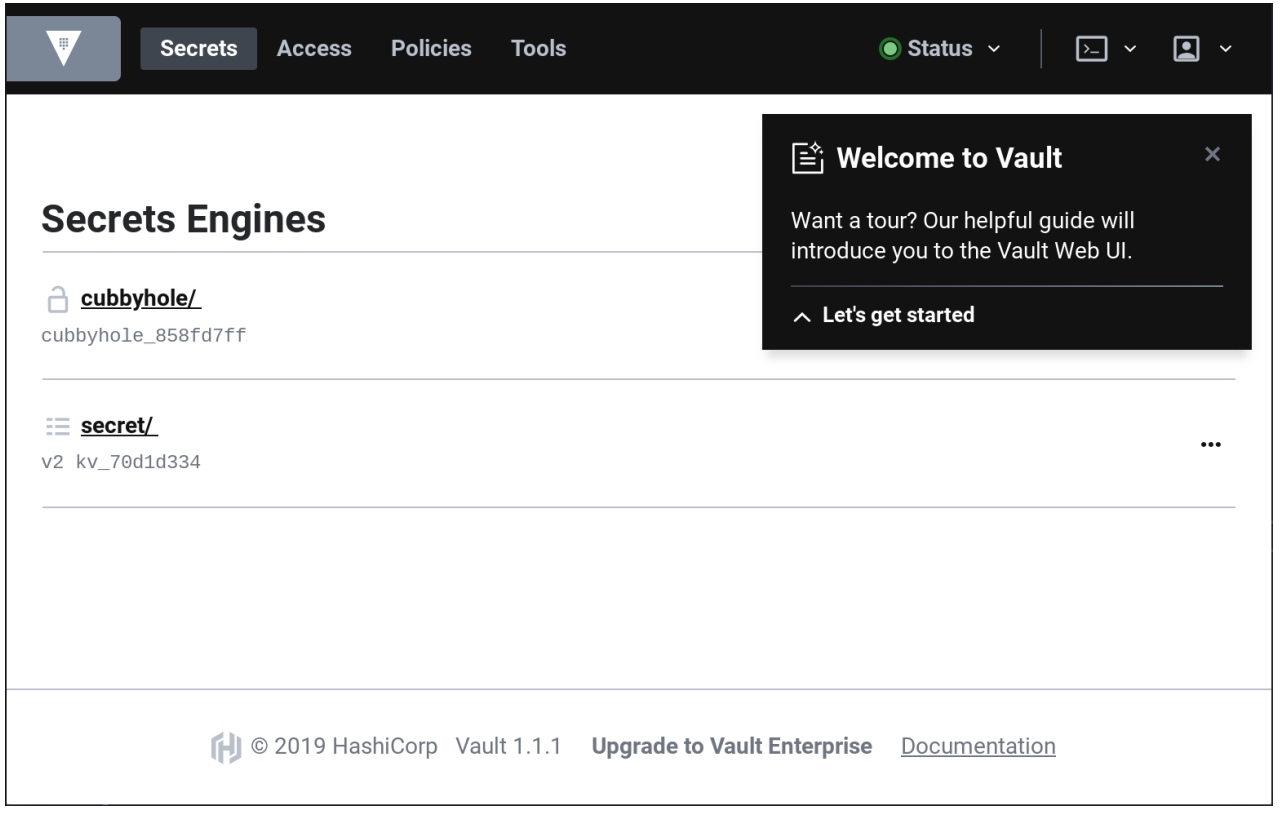
The Vault UI

- In order to use the Vault UI, you must sign in.
- Vault supports multiple authentication methods.
- A new Vault server will only have the Token auth method enabled.
- In the challenge you will soon complete, you use the Token auth method and specify "root" as the token.

Signing into the Vault UI



The "Welcome to Vault" Tour



- When you first login to Vault, you can take a tour.

The Vault API

- Vault has an HTTP API that you can use to configure Vault and manage your secrets.
- You can check Vault's health with a simple `curl` command followed by `jq` to format the JSON output.

Command:

```
curl http://localhost:8200/v1/sys/health | jq
```

The Vault API


```
{
  "initialized": true,
  "sealed": false,
  "standby": false,
  "performance_standby": false,
  "replication_performance_mode": "disabled",
  "replication_dr_mode": "disabled",
  "server_time_utc": 1557180149,
  "version": "1.4.3",
  "cluster_name": "vault-cluster-db6f271d",
  "cluster_id": "33e85d7c-63bb-7523-0165-9d1aee722d70"
}
```

Authenticating Against the Vault API

- The sys/health endpoint didn't require any authentication.
- But most Vault API calls do require authentication.
- This is done with a Vault token that is provided with the `X-Vault-Token` header.

📝 Chapter 2 Review

- How can you interact with Vault?
- What options can you use to get help for Vault commands?
- What are the two Vault server modes?

📝 Chapter 2 Review

- How can you interact with Vault?
  - The Vault CLI
  - The Vault UI
  - The Vault API
- What options can you use to get help for Vault commands?
  - `-h`, `-help`, and `--help`
- What are the two Vault server modes?
  - Dev and Prod

Chapter 3

Running a Production Vault Server

HashiCorp

Running a Production Vault Server

- Running a Vault server in "Prod" mode involves multiple steps:
  - Specify configuration in a config file.
  - Start the server.
  - Initialize the server to get unseal keys and an initial root token.
  - Unseal the Vault server with the unseal keys.

Configuring Vault Servers

- Vault configuration files can be specified in HCL or JSON.
- Common configuration settings include:
  - listener
  - storage
  - seal
  - log_level
  - ui
  - api_addr
  - cluster_addr

Starting a Production Vault Server

- You use the `vault server` command to start a Vault Production server.
- But, you do not use the `-dev` option.

Initializing Vault Clusters

- Recall that a Vault cluster runs multiple Vault servers.
- Each Vault cluster must be initialized once.
- This is done with the `vault operator init` command.
- The number of key shares and the key threshold can be specified with the `-key-shares` and `key-threshold` options.
- The command returns the unseal keys and the initial root token for the cluster.

Unsealing Vault Servers

- Each Vault server must be unsealed each time it is started.
- You cannot use the server until you unseal it.
- This is done with the `vault operator unseal` command, using the unseal keys returned when you initialized the cluster.

Determining the Status of a Vault Server

- Use the `vault status` command to get the status of a Vault server.
- It will tell you if your Vault server is sealed or unsealed.
- It will also tell you the following:
  - the number of key shares and the key threshold
  - whether HA mode (clustering) is enabled
  - whether the server is running as a performance standby.

📝 Chapter 3 Review

- What is used to configure a "Prod" mode Vault server?
- What Vault command needs to be run once against a new Vault cluster?
- What Vault command has to be run each time a Vault server is started?

📝 Chapter 3 Review

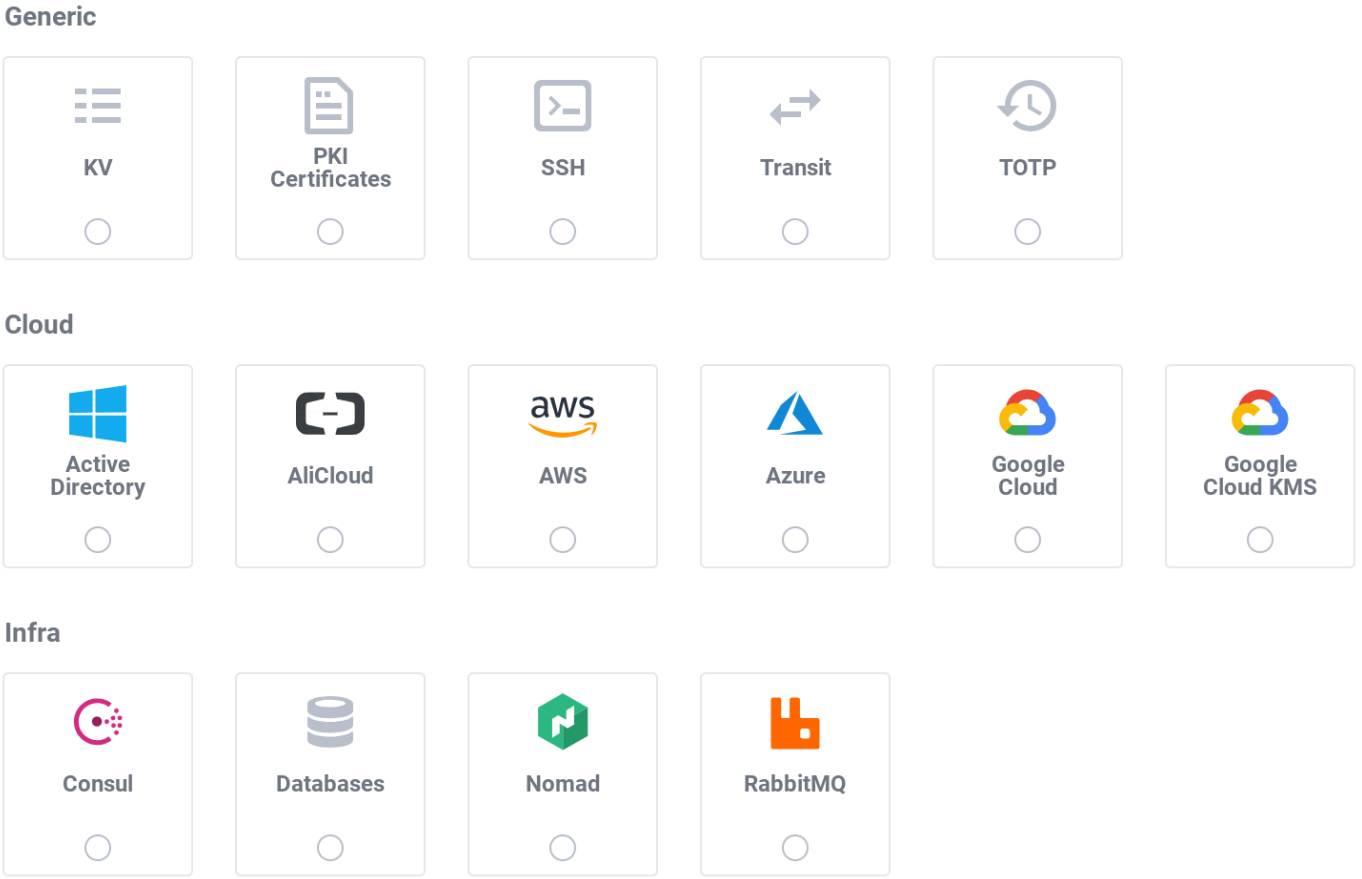
- What is used to configure a "Prod" mode Vault server?
  - A configuration file
- What Vault command needs to be run once against a new Vault cluster?
  - `vault operator init`
- What Vault command has to be run each time a Vault server is started?
  - `vault operator unseal`

Chapter 4

Vault Secrets Engines

HashiCorp

Vault Secrets Engines



Vault includes many different secrets engines.

Important Vault Secrets Engines

- Key/Value (KV)
- PKI
- SSH
- TOTP
- Databases
- AWS, Azure, and Google
- Active Directory
- Transit

Enabling Secrets Engines

- Most Vault secrets engines need to be explicitly enabled.
- This is done with the `vault secrets enable` command.
- Each secrets engine has a default path.
- Alternate paths can be specified to enable multiple instances:

  ```
  vault secrets enable -path=aws-east aws
  ```

- Custom paths must be specified in CLI commands and API calls:

  ```
  vault write aws-east/config/root
  ```

  instead of

  ```
  vault write aws/config/root
  ```

Vault's KV Secrets Engine

- Vault's KV secrets engine actually has 2 versions:
  - KV v1 (without versioning)
  - KV v2 (with versioning)
- In the second lab challenge, we will use the instance of the KV v2 engine that is automatically enabled for "Dev" mode Vault servers.
- Vault does not enable any instances of the KV secrets engine for "Prod" mode servers.
- So, you'll need to enable it yourself.

KV Secrets Engine Commands

- Use this command to mount an instance of the KV v2 secrets engine on the default path `kv`:

  ```
  vault secrets enable -version=2 kv
  ```

- The `vault kv` commands allow you to interact with KV engines.
  - `vault kv list` lists secrets at a specified path.
  - `vault kv put` writes a secret at a specified path.
  - `vault kv get` reads a secret at a specified path.
  - `vault kv delete` deletes a secret at a specified path.
- Other `vault kv` subcommands operate on versions of KV v2 secrets.

📝 Chapter 4 Review

- What option is added to the `vault secrets enable` command to enable multiple instances?
- What is the difference between the two versions of the KV secrets engine?
- Can an old version of a KV v2 secret be retrieved?

📝 Chapter 4 Review

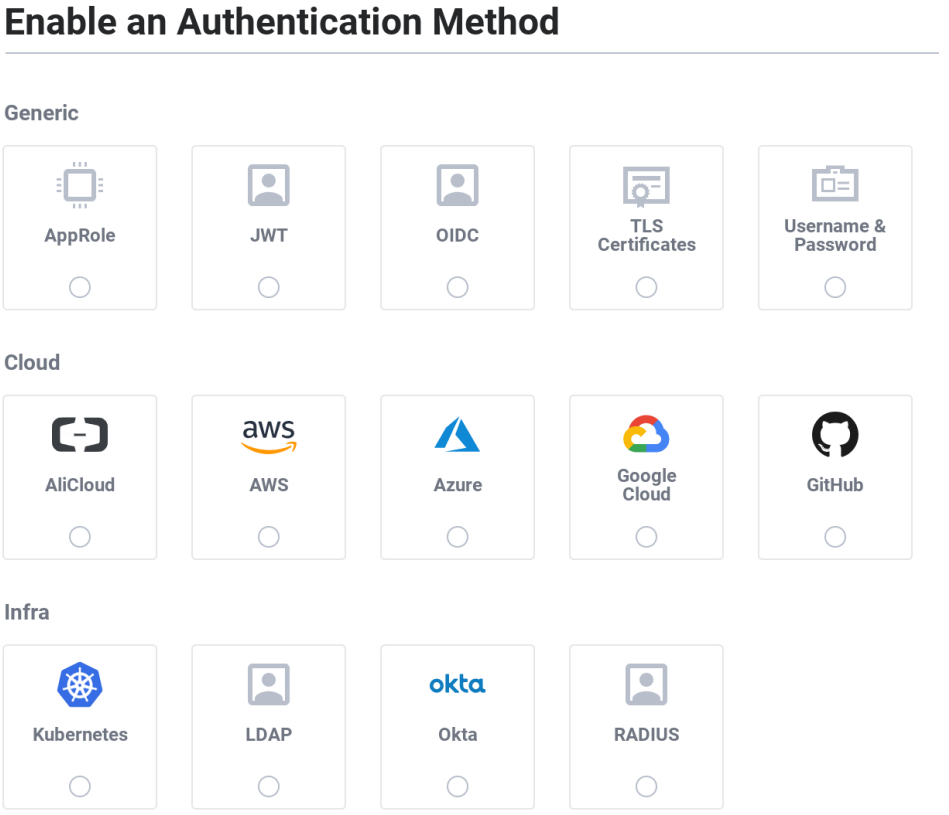
- What option is added to the `vault secrets enable` command to enable multiple instances?
  - Add the `-path=<path>` option and use `<path>` with the CLI and API.
- What is the difference between the two versions of the KV secrets engine?
  - KV V2 supports versioning of secrets.
- Can an old version of a KV v2 secret be retrieved?
  - Yes. You will do this in Vault UI in the challenge.

Chapter 5

Vault Authentication Methods

HashiCorp

Vault Authentication Methods



Vault supports many different authentication methods.

Some Important Vault Auth Methods

Methods for Users

- Userpass
- GitHub
- LDAP
- JWT/OIDC
- Okta

Methods for Applications

- AppRole
- AWS
- Azure
- Google Cloud
- Kubernetes

Enabling Authentication Methods

- Most Vault auth methods need to be explicitly enabled.
- This is done with the `vault auth enable` command.
- Each auth method has a default path.
- Alternate paths can be specified to enable multiple instances:

  ```
  vault auth enable -path=aws-east aws
  ```

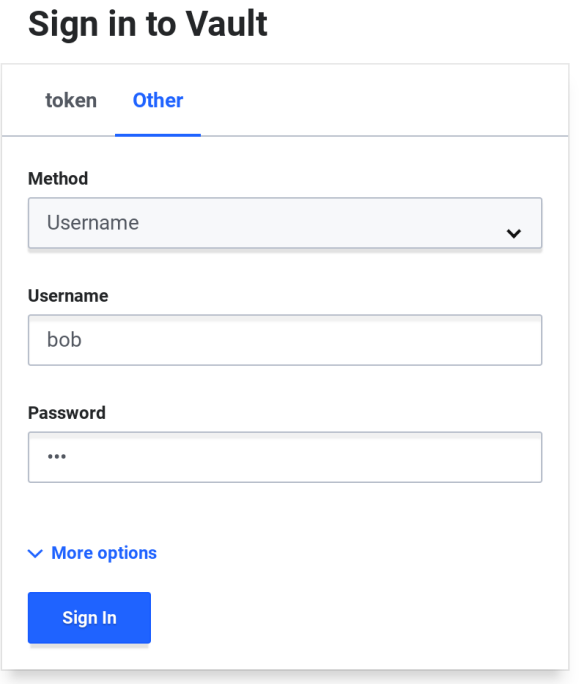
- Custom paths must be specified in CLI commands and API calls:

  ```
  vault write aws-east/config/root
  ```

  instead of

  ```
  vault write aws/config/root
  ```

Vault's Userpass Auth Method



- The Userpass method authenticates users with usernames and passwords managed by Vault.

📝 Chapter 5 Review

- What types of entities can Vault authenticate?
- What system manages credentials for the Userpass auth method?
- Can a user that is not assigned any policies other than the default policy access any secrets?

📝 Chapter 5 Review

- What types of entities can Vault authenticate?
  - Users and applications
- What system manages credentials for the Userpass auth method?
  - Vault
- Can a user that is not assigned any policies other than the default policy access any secrets?
  - No

Chapter 6

Vault Policies

HashiCorp

Vault Policies

- Vault Policies restrict the secrets users and applications have access to.
- Vault follows the practice of least privilege, *denying* access by default.
- Vault administrators must explicity grant users and applications access to specific paths with policy statements.
- In addition to specifying paths, policies also specify a set of capabilities for those paths.
- Policies are written in HashiCorp Configuration Language (HCL).

A Vault Policy Example

- Here is an example of a Vault policy:

```
# Allow tokens to look up their own properties
path "auth/token/lookup-self" {
capabilities = ["read"]
}
```

- Note that this policy does not allow tokens to change their own properties.

Policy Paths and Capabilities

- The path of a policy maps to a Vault API path.
- The most common capabilities granted are: `create`, `read`, `update`, `delete`, and `list` which correspond to HTTP verbs like POST and GET.
- Two other capabilities do not correspond to HTTP verbs:
  - `sudo` allows access to paths that are root-protected.
  - `deny` denies access to a path and takes precedence over other capabilities.

Configuring Policies for LOBs

- Many organizations organize Vault secrets by line of business (LOB) and department.
- Here's an example policy for line of business A, department 1:

```
path "lob_a/dept_1/*" {
    capabilities = ["read", "list", "create", "delete", "update"]
}
```

- This policy grants all standard capabilities to all secrets mounted under `lob_a/dept_1/` by using the glob character (`*`).

Vault Policy CLI Commands

- Vault policies can be added to a Vault server using Vault's CLI, UI, or API.
- The command to add a policy with the CLI is `vault policy write`.
- Here is a command that creates a policy called "lob-A-dept-1" from the HCL file "lob-A-dept-1-policy.hcl":

```
vault policy write lob-A-dept-1 lob-A-dept-1-policy.hcl
```

- Here is a command that associates this policy with a Userpass user:

```
vault write auth/userpass/users/joe/policies policies=lob-A-dept-1
```

📝 Chapter 6 Review

- Does Vault grant access to secrets by default?
- What are the policy capabilities that correspond to HTTP verbs?
- What CLI command can be used to add a policy to Vault?

📝 Chapter 6 Review

- Does Vault grant access to secrets by default?
  - No
- What are the policy capabilities that correspond to HTTP verbs?
  - `create`, `read`, `update`, `delete`, and `list`
- What CLI command can be used to add a policy to Vault?
  - `vault policy write`

Vault Basics Lab

HashiCorp

Doing Labs with Instruqt

- Instruqt is the platform used for HashiCorp workshops.
- Instruqt labs are run in "tracks" that are divided into "challenges".
- If you've never used Instruqt before, start with this tutorial.
- Otherwise, you can skip to the next slide.
- This "Vault Basics" lab covers concepts introduced in chapters 2-6.
- Your instructor will provide links to the tracks.

👩‍💻 Challenge 1: The Vault CLI

- In this lab, you'll run some of the Vault CLI commands.
- You'll do this in the first challenge, "The Vault CLI", of the **Vault Basics** Instruqt track using the link provided by your instructor.

👩‍💻 Challenge 1: Instructions

- Start the "Vault Basics" track by clicking the purple "Start" button on the "Vault CLI" challenge of the track.
- While the challenge is loading, read the displayed text.
- Click the green "Start" button to start the "Vault CLI" challenge.
- Follow the instructions on the right side of the challenge.
- After completing all the steps, click the green "Check" button to see if you did everything right.
- You can also click the "Check" button for reminders.

👩‍💻 Challenge 2: Run a Vault "Dev" Server

- In this lab, you'll run your first Vault server in "Dev" mode.
- You'll also write your first secret to Vault and use the UI.
- Instructions:
  - If the track does not do it for you, click the "Your First Secret" challenge of the "Vault Basics" track.
  - Then click the green "Start" button.
  - Follow the challenge's instructions.
  - Click the green "Check" button when finished.

👩‍💻 Challenge 3: Use the Vault HTTP API

- In this lab, you'll use the Vault HTTP API.
- You'll first check the health of your Vault server.
- You'll then read your `my-first-secret` secret from Vault.
- Instructions:
  - If the track does not do it for you, click the "The Vault API" challenge in the "Vault Basics" track.
  - Then click the green "Start" button.
  - Follow the challenge's instructions.
  - Click the green "Check" button when finished.

👩‍💻 Challenge 4: Run a Vault "Prod" Server

- In this lab, you'll run your first Vault server in "Prod" mode.
- You'll learn how to initialize and unseal a Vault server.
- Instructions:
  - If the track does not do it for you, click the "Run a Production Server" challenge of the "Vault Basics" track.
  - Then click the green "Start" button.
  - Follow the challenge's instructions.
  - Click the green "Check" button when finished.

👩‍💻 Challenge 5: Use the KV v2 Secrets Engine

- In this lab, you'll enable and use the KV v2 secrets engine.
- Note that the path will be `kv` instead of `secret`.
- Instructions:
  - If the track does not do it for you, click the "Use the KV V2 Secrets Engine" challenge of the "Vault Basics" track.
  - Then click the green "Start" button.
  - Follow the challenge's instructions.
  - Click the green "Check" button when finished.

👩‍💻 Challenge 6: Userpass Auth Method

- In this lab, you'll enable and use the Userpass auth method.
- Instructions:
  - If the track does not do it for you, click the "Use the Userpass Auth Method" challenge of the "Vault Basics" track.
  - Then click the green "Start" button.
  - Follow the challenge's instructions.
  - Click the green "Check" button when finished.

👩‍💻 Challenge 7: Vault Policies

- In this lab, you'll use Vault policies to grant different users access to different secrets.
- Instructions:
  - If the track does not do it for you, click the "Use Vault Policies" challenge of the "Vault Basics" track.
  - Then click the green "Start" button.
  - Follow the challenge's instructions.
  - Click the green "Check" button when finished.

Chapter 7

Dynamic Database Secrets

HashiCorp

Dynamic Secrets: Protecting Databases

- Database credentials are normally long-lived.
- Vault's Database Secrets Engine dynamically generates short-lived credentials for databases.
- It supports configuration of database connections and roles with different permissions and time to live (TTL) settings.
- Users or applications request credentials for a specific role from Vault.
- Vault manages the lifecycle of these credentials, automatically deleting them from the database when the TTL expires.

Database Secrets Engine: Plugins

- Cassandra
- Elasticsearch
- Influxdb
- HanaDB
- MongoDB
- MSSQL
- MySQL/MariaDB
- PostgreSQL
- Oracle

Database Secrets Engine Workflow

1. Enable an instance of the database secrets engine.
2. Configure it with the correct plugin and connection URL, using a service account created for Vault.
3. Create one or more roles with TTLs and SQL statements that specify required permissions.
4. Applications and users can request credentials from Vault that are valid for the default TTL of the role, but can be renewed up to the max TTL.
5. Vault automatically deletes expired credentials.
6. If credentials are compromised, you can revoke them immediately.

Lab Environment for Chapters 7 and 8

- In the labs for chapters 7 and 8, we'll use a MySQL database server that runs on the Vault server.
- We will also use a Python web application that stores its data in the MySQL database, but not until chapter 8.
- In the next few slides, we outline many of the steps you will do in the lab.

Configuration Steps for MySQL

1. Enable the database secrets engine on some path.
2. Configure it with the MySQL plugin, connection URL, username, password, and allowed roles.
3. Rotate the "root credentials": Vault modifies the password given in step 2 so that no humans know it anymore.
4. Create roles that can create new credentials that are valid for a specific period of time.

Configuring Connections for MySQL

Run these commands to enable the Database secrets engine and configure a connection for use with MySQL:

```
vault secrets enable -path=lob_a/workshop/database database

vault write lob_a/workshop/database/config/wsmysqldatabase \
    plugin_name=mysql-database-plugin \
    connection_url="{{username}}:{{password}}@tcp(localhost:3306)/" \
    allowed_roles="workshop-app","workshop-app-long" \
    username="hashicorp" \
    password="Password123"

vault write -force lob_a/workshop/database/rotate-root/wsmysqldatabase
```

This creates a connection called "wsmysqldatabase" against the MySQL server on localhost.

Rotating the Root Credentials for MySQL

1. You should not use the actual root user of the database (despite the reference to "root credentials"); instead, create a separate user with sufficient privileges to create users and to change its own password. This can be done by running `GRANT ALL PRIVILEGES on *.* to 'hashicorp'@'%' with grant option;` for the user.

2. The actual username you provide should be for host `'%'`. So, create a user like `'hashicorp'@'%'` rather than `'hashicorp'@'localhost'`.

3. If you don't want to use `'%'` as the host for the user, you can specify `root_rotation_statements` when writing to the path `<database>/config/<connection>`; for instance, you could set this to `"ALTER USER '{{username}}'@'localhost' IDENTIFIED BY '{{password}}';"`.

Configuring Roles for MySQL

Run this command to configure a role for MySQL:

```
vault write lob_a/workshop/database/roles/workshop-app-long \
    db_name=wsmysqldatabase \
    creation_statements="CREATE USER '{{name}}'@'%' IDENTIFIED BY '{{password}}';
    GRANT ALL ON my_app.* TO '{{name}}'@'%';" \
    default_ttl="1h" \
    max_ttl="24h"
```

This defines a role against the "wsmysqldatabase" connection which generates credentials with an initial TTL of 1 hour. But their lifetime can be extended up to 24 hours.

Generating Database Credentials

Run this command to generate actual credentials for the MySQL database against the role that was configured on the previous slide:

```
vault read lob_a/workshop/database/creds/workshop-app-long
```

This should return something like:

```
Key                Value
---                -----
lease_id           lob_a/workshop/database/creds/workshop-app-long/JeUGIL2xD6BzXSneqity8UmF
lease_duration     1h
lease_renewable    true
password           A1a-zy4ENaf2kwpzGk9t
username           v-token-workshop-a-DM0BJ3eMlMhbf
```

Renewing and Revoking Database Credentials

Run this command to renew credentials, replacing `<lease_id>` with the right lease_id:

```
vault write sys/leases/renew lease_id="<lease_id>" increment="120"
```

Run this command to revoke credentials, replacing `<lease_id>` with the right lease_id:

```
vault write sys/leases/revoke lease_id="<lease_id>"
```

You can also determine the remaining lifetime of the credentials:

```
vault write sys/leases/lookup lease_id="<lease_id>"
```

👩‍💻 Challenge 1: Enable the Engine

- In this lab challenge, you'll enable the database engine for MySQL and rotate its root credentials.
- You'll do this in the **Vault Dynamic Database Credentials** Instruqt track using the link provided by your instructor.
- Instructions:
  - Click the "Enable the Database Secrets Engine" challenge of the "Vault Dynamic Database Credentials" track.
  - Then click the green "Start" button.
  - Follow the challenge's instructions.
  - Click the green "Check" button when finished.

👩‍💻 Challenge 2: Configure the Engine

- In this lab, you'll configure a connection and two roles for the database.
- Instructions:
  - If the track does not do it for you, click the "Configure the Database Secrets Engine" challenge of the "Vault Dynamic Database Credentials" track.
  - Then click the green "Start" button.
  - Follow the challenge's instructions.
  - Click the green "Check" button when finished.

👩‍💻 Challenge 3: Generate Credentials

- In this lab, you'll generate and use credentials against both roles that you configured in the previous challenge.
- Instructions:
  - If the track does not do it for you, click the "Generate and Use Dynamic Database Credentials" challenge of the "Vault Dynamic Database Credentials" track.
  - Then click the green "Start" button.
  - Follow the challenge's instructions.
  - Click the green "Check" button when finished.

🤦 Challenge 4: Renew and Revoke Credentials

- In this lab, you'll renew and revoke credentials generated by the database secrets engine.
- Instructions:
  - If the track does not do it for you, click the "Renew and Revoke Database Credentials" challenge of the "Vault Dynamic Database Credentials" track.
  - Then click the green "Start" button.
  - Follow the challenge's instructions.
  - Click the green "Check" button when finished.

📝 Chapter 7 Review

- What is the main advantage of using Vault's database secrets engine?
- What happens when credentials expire?
- Is the database engine limited to the plugins listed in this chapter?
- Can more than one role be used against a single connection?

📝 Chapter 7 Review

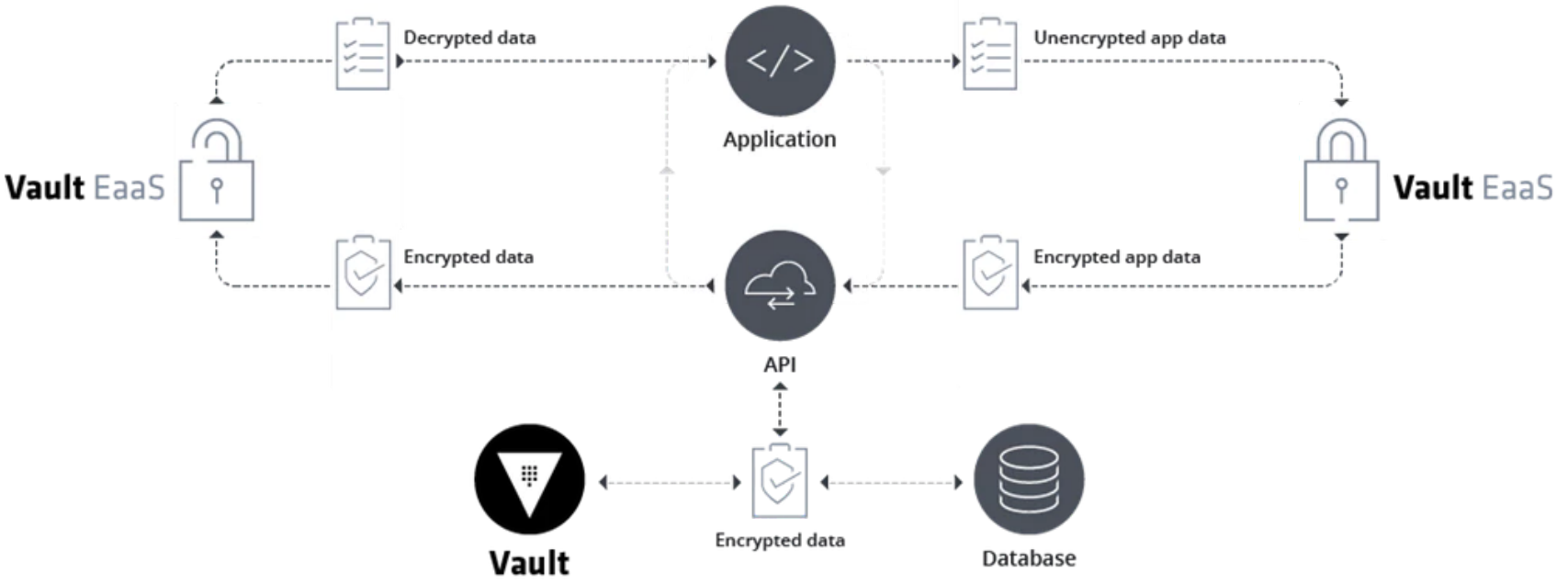
- What is the main advantage of using Vault's database secrets engine?
  - The credentials are short-lived and less likely to be compromised.
- What happens when credentials expire?
  - Vault deletes them from the database server.
- Is the database engine limited to the plugins listed in this chapter?
  - No. Custom plugins can be written.
- Can more than one role be used against a single connection?
  - Yes. This allows different apps to get credentials with different TTLs.

Chapter 8

Encryption as a Service

HashiCorp

Vault Transit Engine - Encryption as a Service



- Vault's Transit Secrets Engine functions as an Encryption-as-a-Service.
- Developers use it to encrypt and decrypt data stored outside of Vault.

Transit Engine Benefits

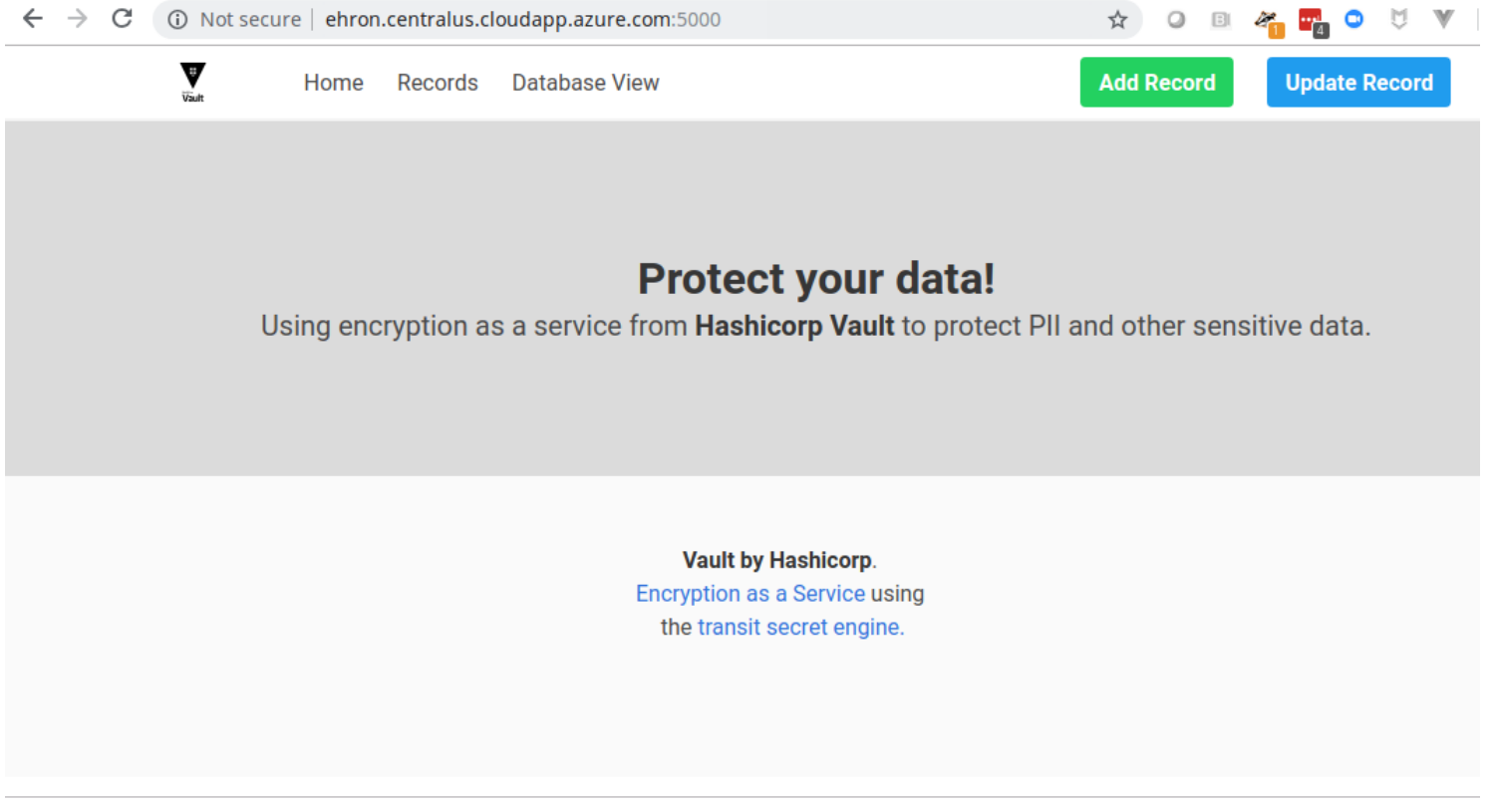
- Vault's Transit Engine provides developers a well-architected EaaS API so that they don't have to become encryption or cryptography experts.
- It provides centralized key management.
- It ensures that only approved ciphers and algorithms are used.
- It supports automated key rotation and re-wrapping.
- If an attacker manages to get access to the encrypted data, they will only see ciphertext that is useless without Vault.

Vault Transit - Example Application

- In the next lab we'll use a web application that uses the Transit engine to encrypt and decrypt data.
- The app will store its encrypted data in the same MySQL database we used in Chapter 7.
- It will also get MySQL credentials from the Database secrets engine we configured in that chapter's lab.
- We'll first run the web app without Vault: No records are encrypted.
- We'll then run it with Vault enabled and see that new records are encrypted.

The Web App

Here is a screenshot of the Python web app:

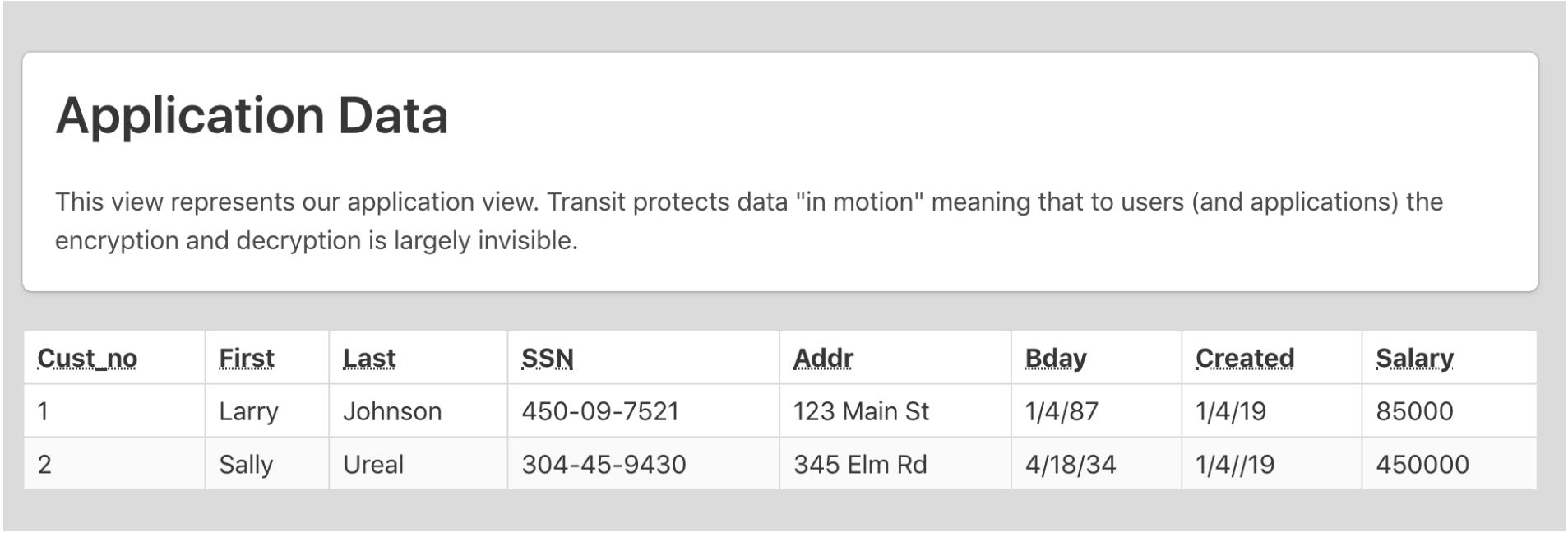


The Web App's Views

There are two main sections in the application.

1. **Records View**
    - The Records View displays records in plain text, showing what a logged in user would see after any encrypted data is decrypted.
2. **Database View**
    - The Database View displays the raw records in the database, showing what SQL commands would return:

The Web App's Records View



Application Data

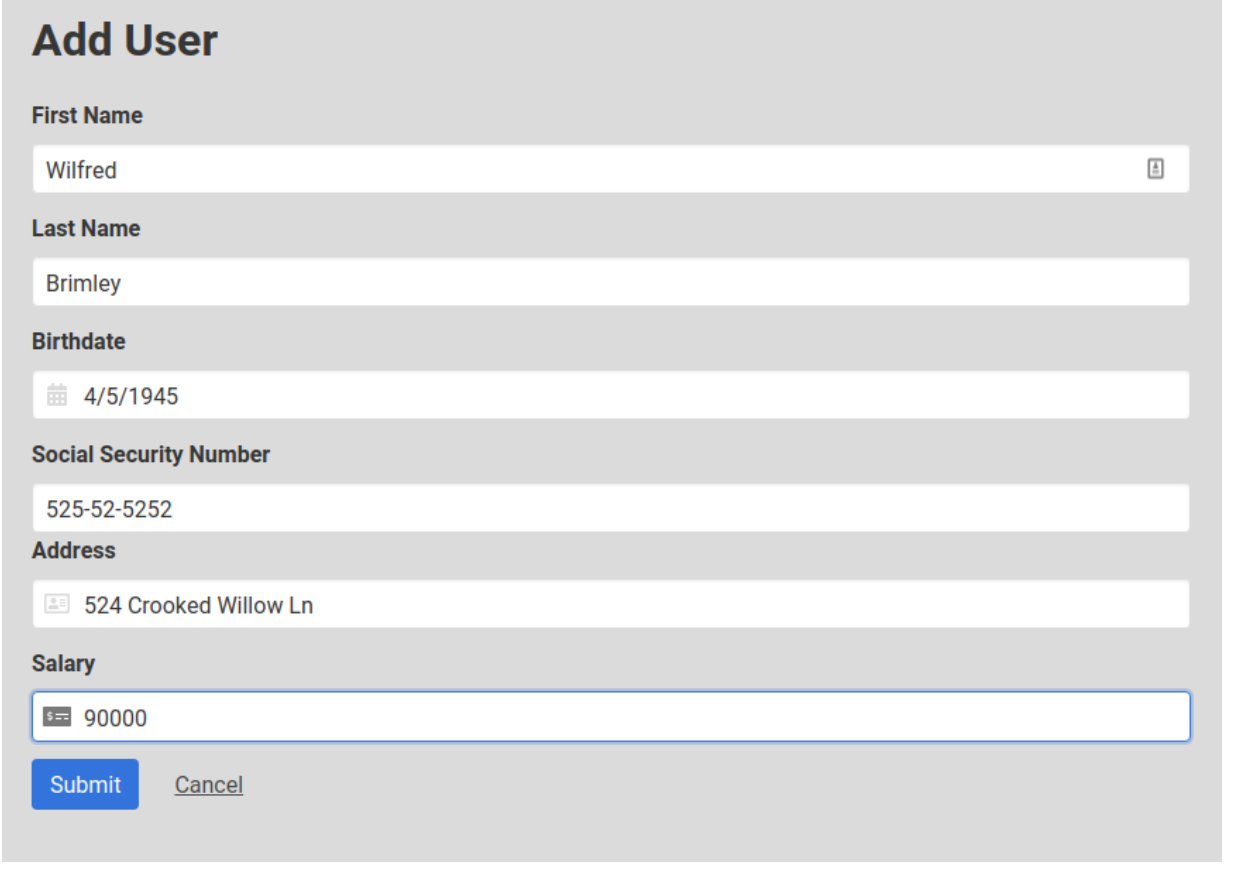
This view represents our application view. Transit protects data "in motion" meaning that to users (and applications) the encryption and decryption is largely invisible.

Cust_no	First	Last	SSN	Addr	Bday	Created	Salary
1	Larry	Johnson	450-09-7521	123 Main St	1/4/87	1/4/19	85000
2	Sally	Ureal	304-45-9430	345 Elm Rd	4/18/34	1/4//19	450000

- As we would expect an authorized user is able to see some of the sensitive data because the app has decrypted any encrypted data.

The Add User Screen

- In the lab, you will add new users to the database.

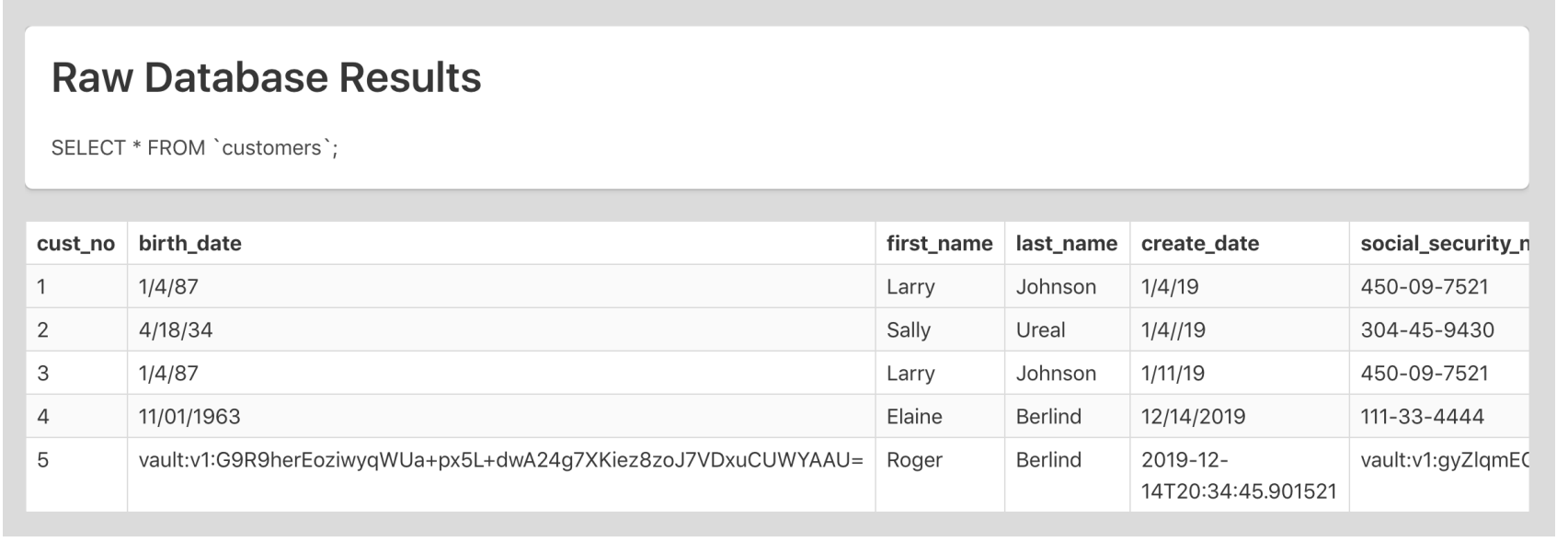


Record in Database View Without Vault Enabled

- After adding a record in the lab, you will be instructed to click on the **Database View** menu.
- You should see the exact same data that you entered.
- This means that Personally Identifiable Data (PII) is being stored in plain text in our database records.
- How can we improve this? Let's enable Vault's Transit engine and see.

A Database Record Encrypted by Vault

Here is a record that was encrypted by Vault's Transit engine.



Raw Database Results

SELECT * FROM `customers`;

cust_no	birth_date	first_name	last_name	create_date	social_security_n
1	1/4/87	Larry	Johnson	1/4/19	450-09-7521
2	4/18/34	Sally	Ureal	1/4//19	304-45-9430
3	1/4/87	Larry	Johnson	1/11/19	450-09-7521
4	11/01/1963	Elaine	Berlind	12/14/2019	111-33-4444
5	vault:v1:G9R9herEoziwyqWUa+px5L+dwA24g7XKiez8zoJ7VDxuCUWYAAU=	Roger	Berlind	2019-12-14T20:34:45.901521	vault:v1:gyZlqmEC

- Note that the birth_date and social_security_number are encrypted.

Rotating Transit Engine Encryption Keys

- The encryption keys of Vaults Transit Engine can be rotated.
- The newest version of the key is used to encrypt new data
- Older versions of the key can still decrypt old data but cannot decrypt new data.
- When we rotate the encryption keys, apps that use the Transit engine are unaware of any changes.
- Data can also be re-encrypted using the `rewrap` endpoint.

👩‍💻 Challenge 1: Enable the Transit Engine

- In this lab challenge, you'll enable the Transit engine.
- You'll do this in the **Vault Encryption as a Service** Instruqt track using the link provided by your instructor.
- Instructions:
  - Click the "Enable the Transit Secrets Engine" challenge of the "Vault Encryption as a Service" track.
  - Then click the green "Start" button.
  - Follow the challenge's instructions.
  - Click the green "Check" button when finished.

👩‍💻 Challenge 2: Create an Encryption Key

- In this lab, you'll create an encryption key for use with the Transit engine you enabled in the previous challenge.
- Instructions:
  - If the track does not do it for you, click the "Create a Key for the Transit Secrets Engine" challenge of the "Vault Encryption as a Service" track.
  - Then click the green "Start" button.
  - Follow the challenge's instructions.
  - Click the green "Check" button when finished.

👩‍💻 Challenge 3: Use the Web App Without Vault

- In this lab, you'll use the web application without Vault.
- Instructions:
  - If the track does not do it for you, click the "Use the Web App Without Vault" challenge of the "Vault Encryption as a Service" track.
  - Then click the green "Start" button.
  - Follow the challenge's instructions.
  - Click the green "Check" button when finished.

👩‍💻 Challenge 4: Use the Web App With Vault

- In this lab, you'll use the web application with Vault.
- You'll also rotate the encryption key.
- Instructions:
  - If the track does not do it for you, click the "Use the Web App With Vault" challenge of the "Vault Encryption as a Service" track.
  - Then click the green "Start" button.
  - Follow the challenge's instructions.
  - Click the green "Check" button when finished.

📝 Chapter 8 Review

- What is the main advantage of using Vault's Transit secrets engine?
- Where does Vault's Transit Engine store encrypted data?
- Was the application still able to decrypt older encrypted records after you rotated the encryption key?
- Is it possible to tell which version of an encryption key was used?

📝 Chapter 8 Review

- What is the main advantage of using Vault's Transit secrets engine?
  - Developers can encrypt data without being experts in cryptography.
- Where does Vault's Transit Engine store encrypted data?
  - Wherever developers want, but outside of Vault
- Was the application still able to decrypt older encrypted records after you rotated the encryption key?
  - Yes
- Is it possible to tell which version of an encryption key was used?
  - Yes. The version is indicated by `v1`, `v2`, etc.

Thank You for Participating!

For more information please refer to the following links:

- https://www.vaultproject.io/docs/
- https://www.vaultproject.io/api/
- https://learn.hashicorp.com/vault

Workshop Feedback Survey

- Your feedback is important to us!
- The survey is short, we promise:
  - http://bit.ly/hashiworkshopfeedback