Enable and Configure Secrets Engines

Vault

CERTIFIED
OPERATIONS
PROFESSIONAL

Available Secrets Engines

Vault
CERTIFIED
OPERATIONS
PROFESSIONAL

- Active Directory
- AliCloud
- AWS
- Azure
- Consul
- Cubbyhole
- Database
- Google Cloud
- Google Cloud KMS
- Key Management
- KMIP
- KV
- Identity
- MongoDB Atlas
- Nomad
- OpenLDAP
- PKI
- RabbitMQ
- SSH
- Terraform Cloud
- TOTP
- Transform
- Transit
- Venafi

Generic Secrets Engines

- HashiCorp can't expect everybody is proficient on all platforms or cloud providers
- These are the secrets engines you'll most likely be tested on

Cubbyhole

Identity

Database

PKI

KV

Transit

Generic Secrets Engines

Vault CERTIFIED OPERATIONS PROFESSIONAL

- The database secrets engine supports 13+ different database platforms using plugins
- Key/Value has two versions, KV v1 and KV v2
- PKI creates X.509 certificates
- Transit encrypts data with encryption keys. Can also be used for Transit auto-unseal

Enabling Secrets Engines

- Cubbyhole and Identity are enabled by default (can’t disable)
- Any other secrets engine must be enabled
  - Can enable using the CLI, API, or UI (most)

- Secrets engines are enabled and isolated at a unique path
  - All interactions with the secrets engine are done using the path
  - Paths do not need to match the secrets engines name or type
    - Make them meaningful for you and your organization

Enabling Secrets Engines

Command Line Interface (CLI)

Use the `vault secrets` command

- `disable`
- `enable`
- `list`
- `move`
- `tune`

Terminal

```
$ vault secrets enable aws
Success! Enabled the aws secrets engine at: aws/
```

Terminal

```
$ vault secrets tune -default-lease-ttl=72h pki/
```

Enabling Secrets Engines

Command Line Interface (CLI)

Helpful flags to use with the `vault secrets` command

- `vault secrets list --detailed`
- `vault secrets enable -path=developers kv`
- `vault secrets enable -description="my first kv" kv`

Terminal

```
$ vault secrets enable -description="My Secrets" -path="cloud-kv" kv-v2
```

Enabling Secrets Engines

Command Line Interface (CLI)

$$\texttt{vault}\ \underbrace{\texttt{secrets}}_{\text{Type of Vault object you want to work with}}\ \underbrace{\texttt{enable}}_{\text{Subcommand}}\ \underbrace{\texttt{-path=bryan}}_{\text{Define a custom path to enable the secrets engine on}}\ \underbrace{\texttt{kv-v2}}_{\text{The type of secrets engine you want to enable}}$$

Enabling Secrets Engines

Command Line Interface (CLI)

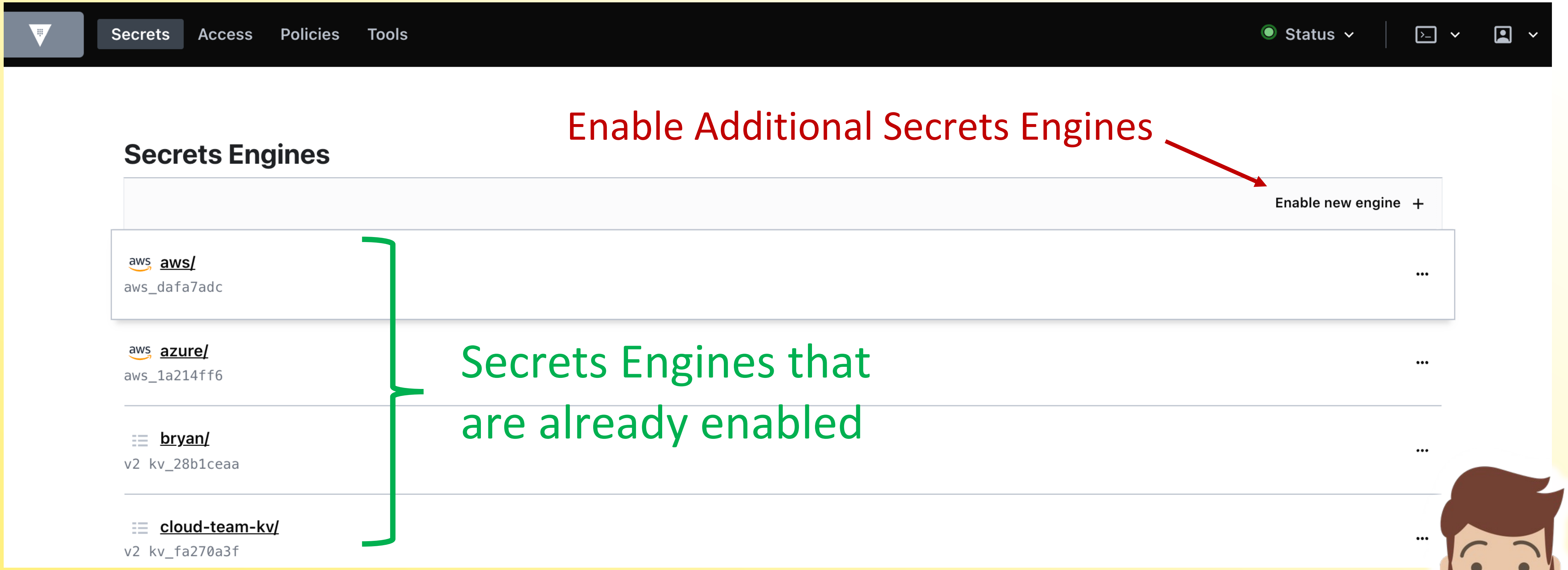
Use the `vault secrets list` command

Terminal

```
$ vault secrets list

Path               Type         Accessor               Description
----               ----         --------               -----------
aws/               aws          aws_dafa7adc           n/a
azure/             aws          aws_1a214ff6           n/a
vault-ops-pro/     kv           kv_28b1ceaa            Earn Your HCVOP Certification
cloud-team-kv/     kv           kv_fa270a3f            n/a
cubbyhole/         cubbyhole    cubbyhole_88c8e2e3     per-token private secret storage
dev-team-kv/       kv           kv_55c319c4            n/a
identity/          identity     identity_e60e93cb      identity store
kv-v2/             kv           kv_eea3206c            n/a
sys/               system       system_66b0d8ee        system endpoints used fo      rol
transit/           transit      transit_7b8038ca       n/a
```

Enabling Secrets Engines

User Interface (UI)



Enable Additional Secrets Engines

Secrets Engines

Enable new engine +

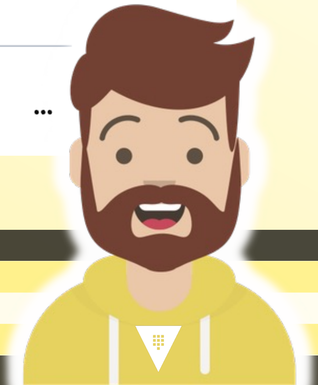
aws aws/
aws_dafa7adc

aws azure/
aws_1a214ff6

bryan/
v2 kv_28b1ceaa

cloud-team-kv/
v2 kv_fa270a3f

Secrets Engines that are already enabled



Vault
CERTIFIED
OPERATIONS
PROFESSIONAL

Key/Value Secrets Engine

Key/Value Secrets Engine

- Key/Value secrets engine is used to store static secrets
  - There are two versions: v2 (kv-v2) is versioned but v1 (v1) is not
  - Secrets are accessible via UI, CLI, and API – interactive or automated
  - Access to KV paths are enforced via policies (ACLs)

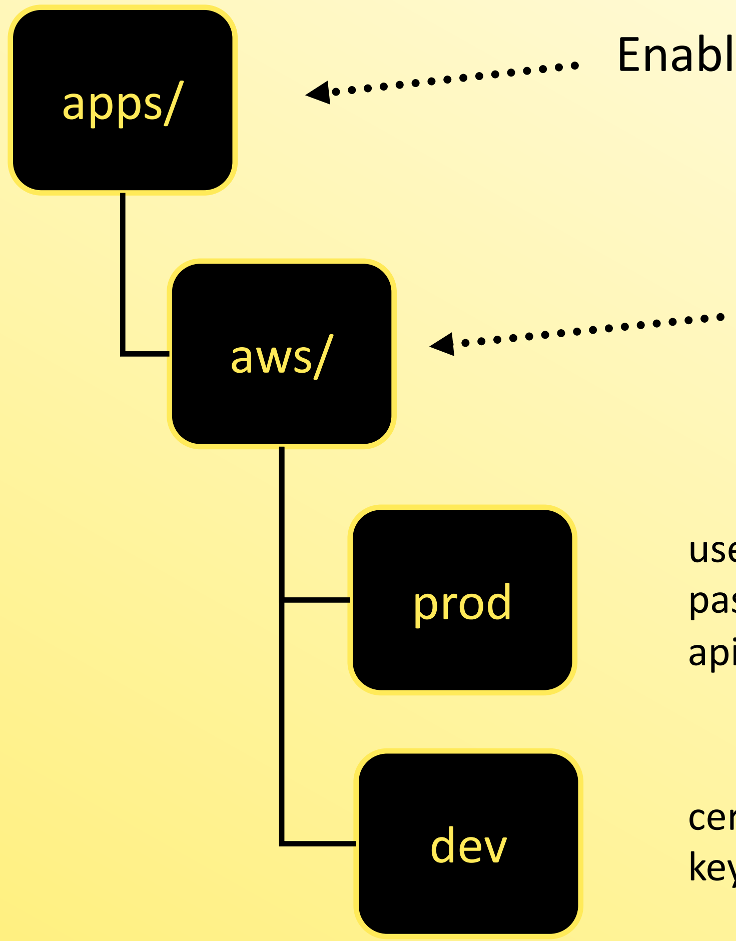
- Like everything else in Vault, secrets written to the KV secrets engine are encrypted using 256-bit AES

Key/Value Secrets Engine

- Key/Value secrets engine can be enabled at different paths
  - Each key/value secrets engine is isolated and unique

- Secrets are stored as key-value pairs at a defined path – (e.g., secret/applications/web01)
  - Writing a new secret will ***replace the old value***
  - Writing a new secret requires the `create` capability
  - Updating/overwriting a secret to an existing path requires `update` capability

Key/Value Secrets Engine



Enabled KV secrets engine at apps/

Just part of a path – currently no secrets stored here but you could if you wanted

user=dbadmin
password=P@ssw0rd
api=b93md83mdmapw

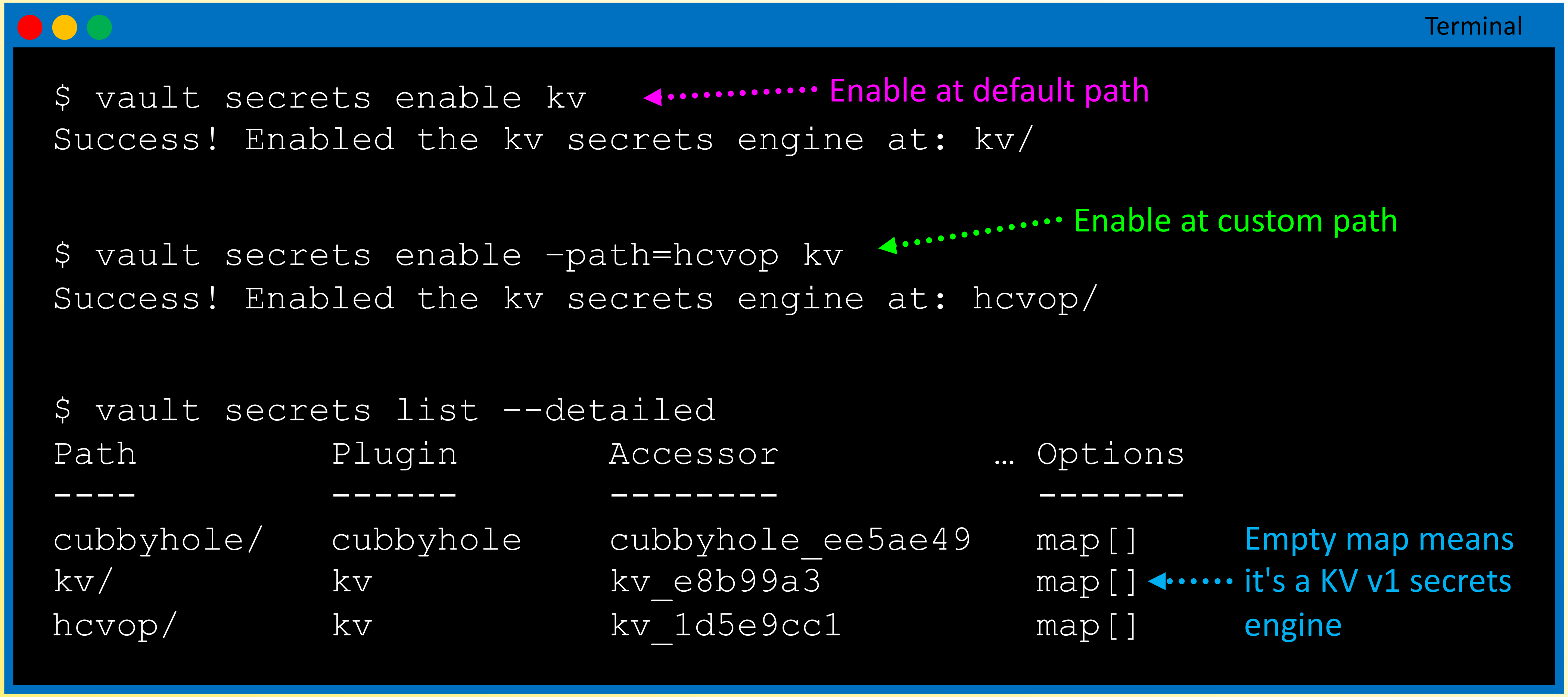
path = apps/aws/prod

cert=---BEGIN CERTI…
key=---BEGIN PRIVA…

path = apps/aws/dev

Key/Value Secrets Engine

Enable Version 1



```
$ vault secrets enable kv
Success! Enabled the kv secrets engine at: kv/


$ vault secrets enable -path=hcvop kv
Success! Enabled the kv secrets engine at: hcvop/


$ vault secrets list --detailed
Path          Plugin       Accessor              … Options
----          ------       --------                -------
cubbyhole/    cubbyhole    cubbyhole_ee5ae49       map[]
kv/           kv           kv_e8b99a3              map[]
hcvop/        kv           kv_1d5e9cc1             map[]
```

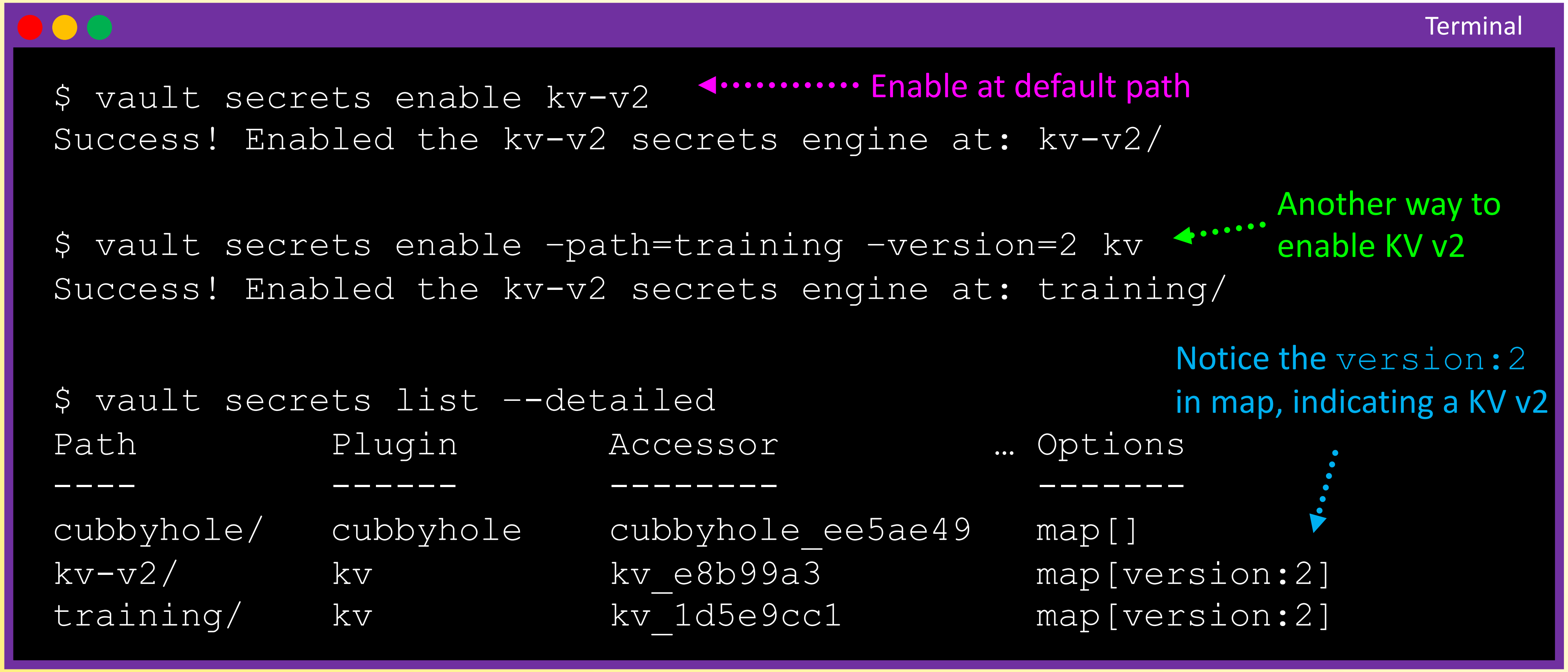
Enable at default path

Enable at custom path

Empty map means it's a KV v1 secrets engine

Key/Value Secrets Engine

Enable KV Version 2



```
$ vault secrets enable kv-v2
Success! Enabled the kv-v2 secrets engine at: kv-v2/

$ vault secrets enable -path=training -version=2 kv
Success! Enabled the kv-v2 secrets engine at: training/

$ vault secrets list --detailed
Path           Plugin       Accessor            … Options
----           -------      ---------             -------
cubbyhole/     cubbyhole    cubbyhole_ee5ae49     map[]
kv-v2/         kv           kv_e8b99a3            map[version:2]
training/      kv           kv_1d5e9cc1           map[version:2]
```

Enable at default path

Another way to enable KV v2

Notice the `version:2` in map, indicating a KV v2

Upgrade KV v1 to KV v2

- You can upgrade a KV v1 secrets engine to a KV v2
- You can't undo this upgrade
- ...and no....you can't go from KV v2 to KV v1

```
$ vault kv enable-versioning training/
Success! Tuned the secrets engine at: training/
```

How is KV v2 Different?

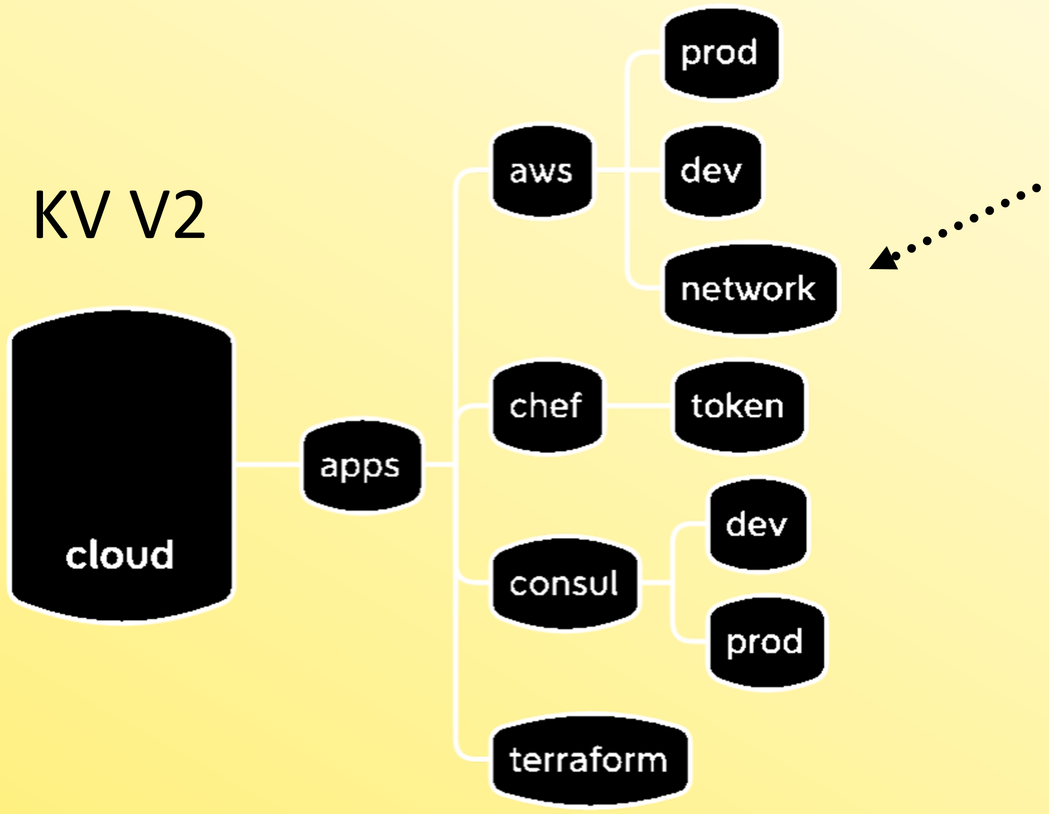
- To support versioning, KV V2 adds metadata to our Key Value entries
- Used to determine creation date, the version of the secret, etc.

- Introduces two prefixes that must be accounted for when referencing secrets and/or metadata

  - `cloud/data` – data is where the actual K/V data is stored
  - `cloud/metadata` – the metadata prefix stores our metadata about a secret

How is KV v2 Different?

KV V2



path = cloud/data/apps/aws/network

data/ prefix to read secrets

Important Operational Concept about KV V2!

How is KV v2 Different?

- The `data/` and `metadata/` prefix is <u>required</u> for API and when writing Vault policies
- It does NOT change the way you interact with the KV v2 store when using the CLI

Working with the Key/Value Secrets Engine

Working with KV using the CLI

Use the `vault kv` command

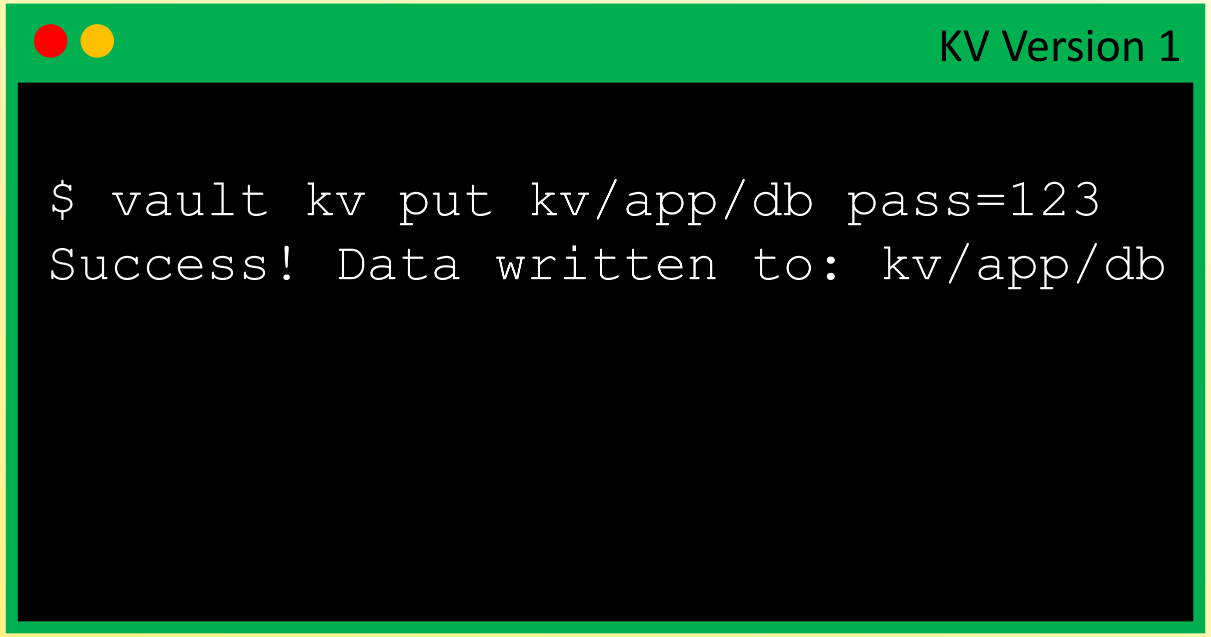
`put`	- write data to the KV
`get`	- read data from the KV
`delete`	- delete data from the KV
`list`	- list data within the KV

`undelete`	- undelete version of secret
`destroy`	- permanently destroy data
`patch`	- add specific key in the KV
`rollback`	- recover old data in the KV

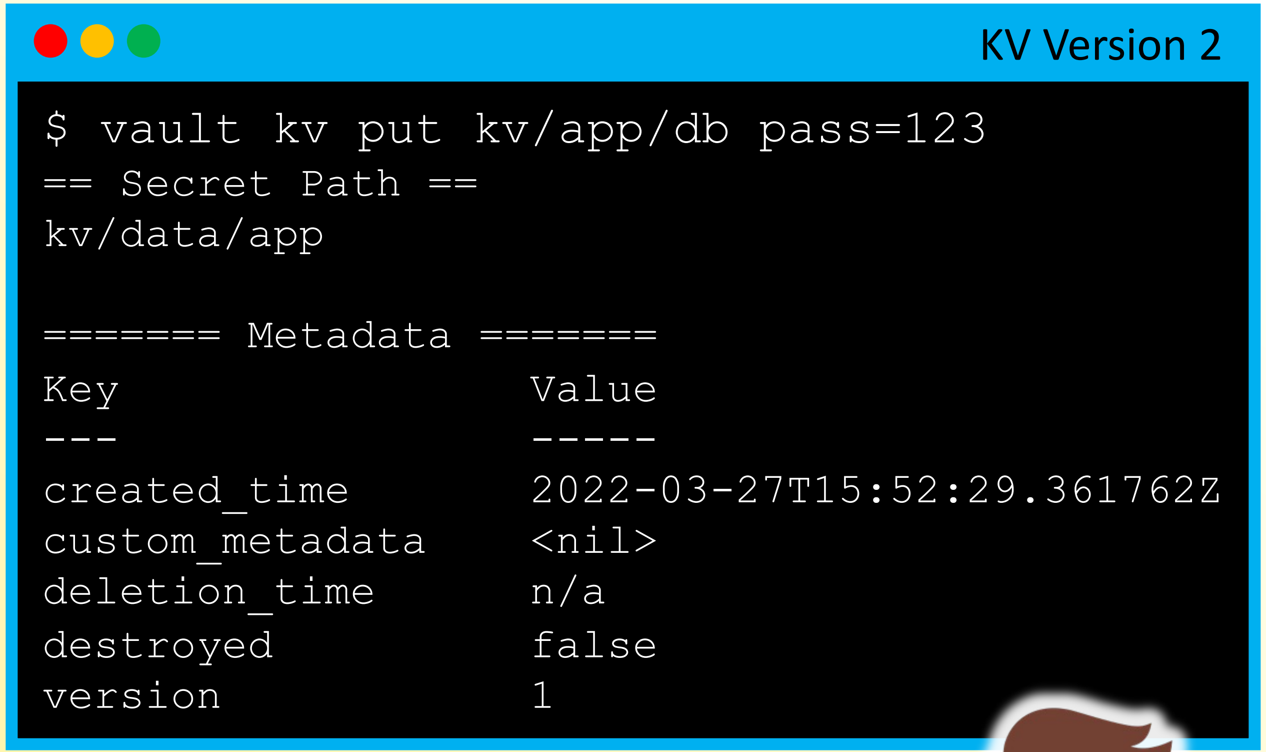
$\left.\begin{array}{l}\text{undelete} \\ \text{destroy} \\ \text{patch} \\ \text{rollback}\end{array}\right\}$ Only available for KV V2

Working with KV using the CLI

Compare KV Version 1 and Version 2


KV Version 1

```
$ vault kv put kv/app/db pass=123
Success! Data written to: kv/app/db
```


KV Version 2

```
$ vault kv put kv/app/db pass=123
== Secret Path ==
kv/data/app

======= Metadata =======
Key                Value
---                -----
created_time       2022-03-27T15:52:29.361762Z
custom_metadata    <nil>
deletion_time      n/a
destroyed          false
version            1
```

The CLI command is the same, but we get different output behavior

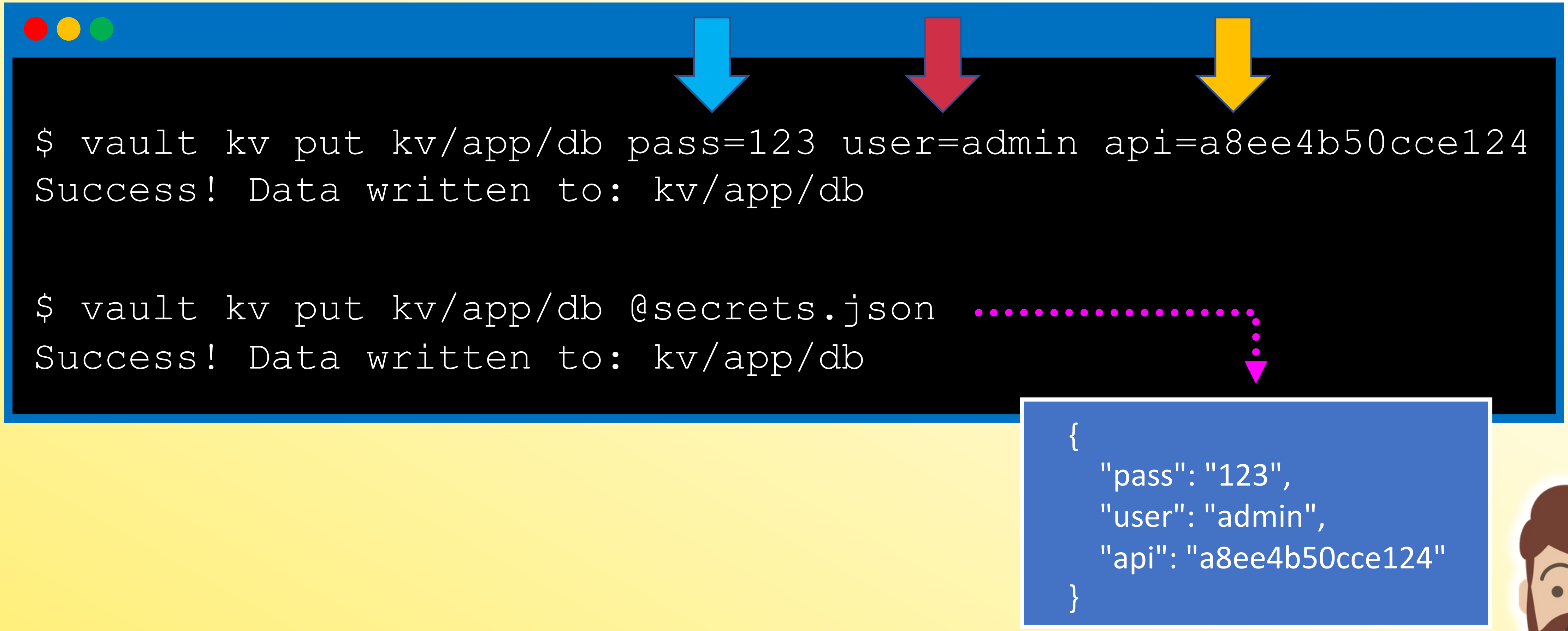
Writing Data to the KV Store

Vault
CERTIFIED
OPERATIONS
PROFESSIONAL

$$\texttt{vault}\ \underbrace{\texttt{kv}}_{\text{Command when working with the KV Store}}\ \underbrace{\texttt{put}}_{\text{Sub-command}}\ \underbrace{\texttt{kv/app/db}}_{\text{Path where you want to store the KV pair}}\ \underbrace{\texttt{pass=123}}_{\text{Data to store – entered as KV pairs}}$$

Writing Data to the KV Store

What if I have a bunch of key pairs?



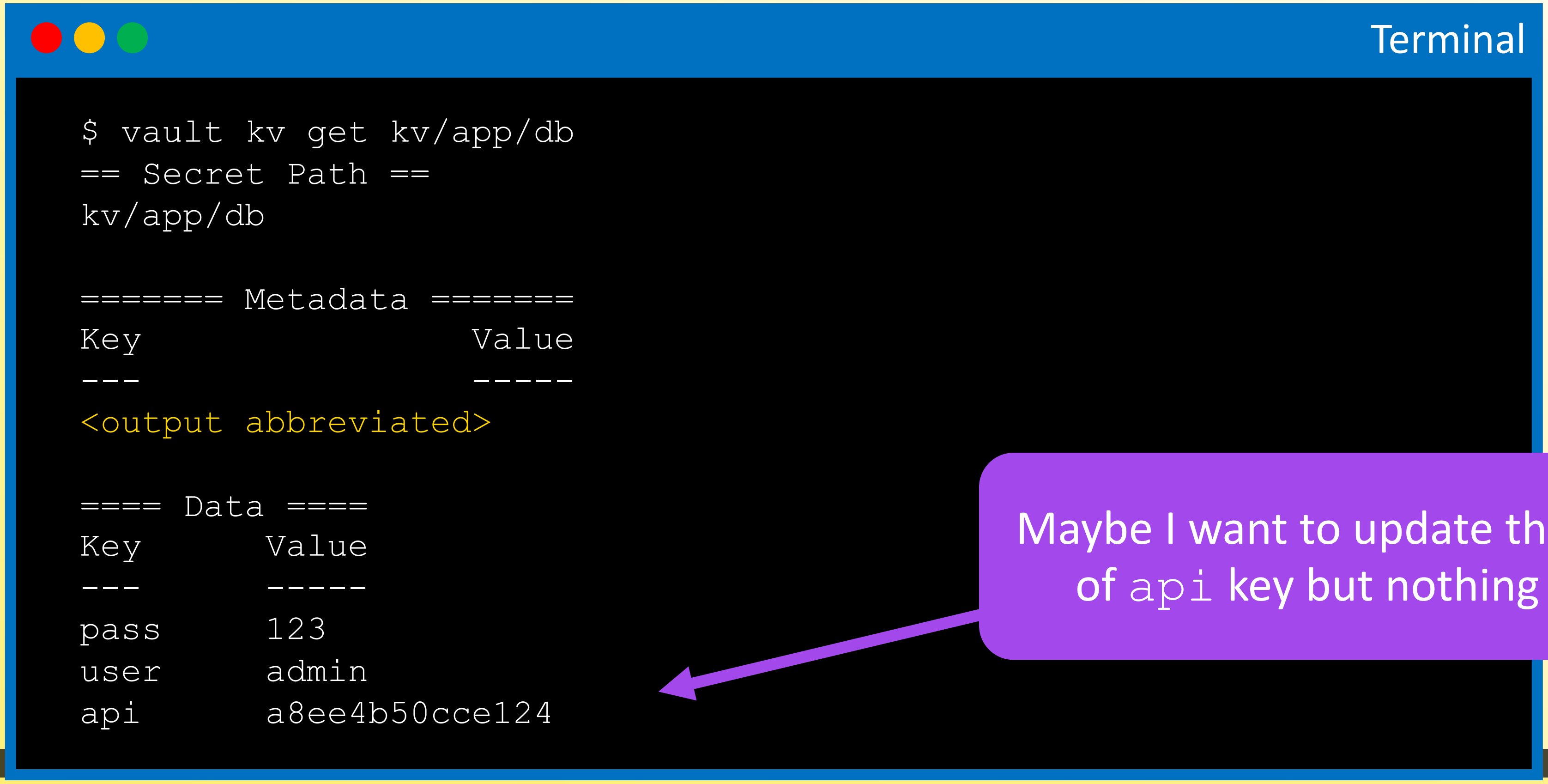
```
$ vault kv put kv/app/db pass=123 user=admin api=a8ee4b50cce124
Success! Data written to: kv/app/db


$ vault kv put kv/app/db @secrets.json
Success! Data written to: kv/app/db
```

```
{
    "pass": "123",
    "user": "admin",
    "api": "a8ee4b50cce124"
}
```

Writing Data to the KV Store

Critical Things to Remember

Writing a new secret will replace the old value

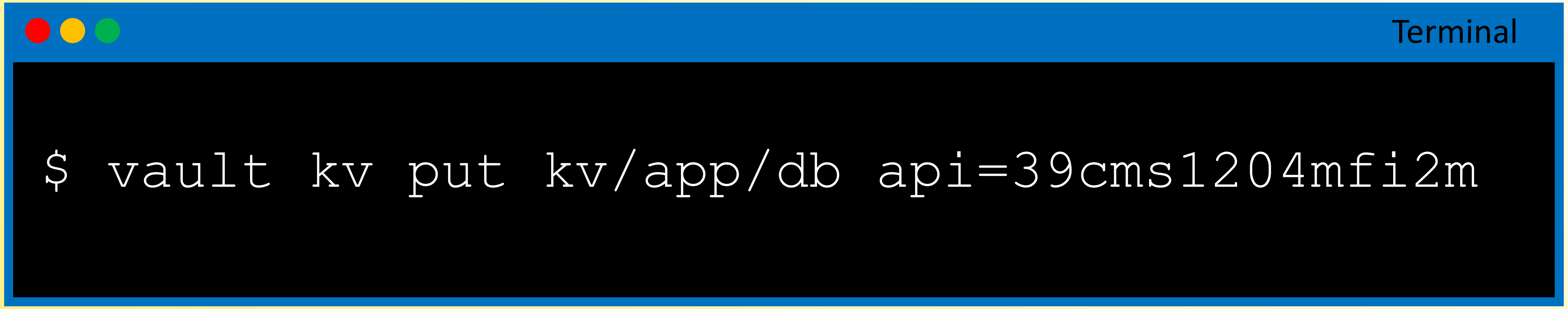

Terminal

```
$ vault kv get kv/app/db
== Secret Path ==
kv/app/db

======= Metadata =======
Key                  Value
---                  -----
<output abbreviated>

==== Data ====
Key        Value
---        -----
pass       123
user       admin
api        a8ee4b50cce124
```

Maybe I want to update the value of `api` key but nothing else

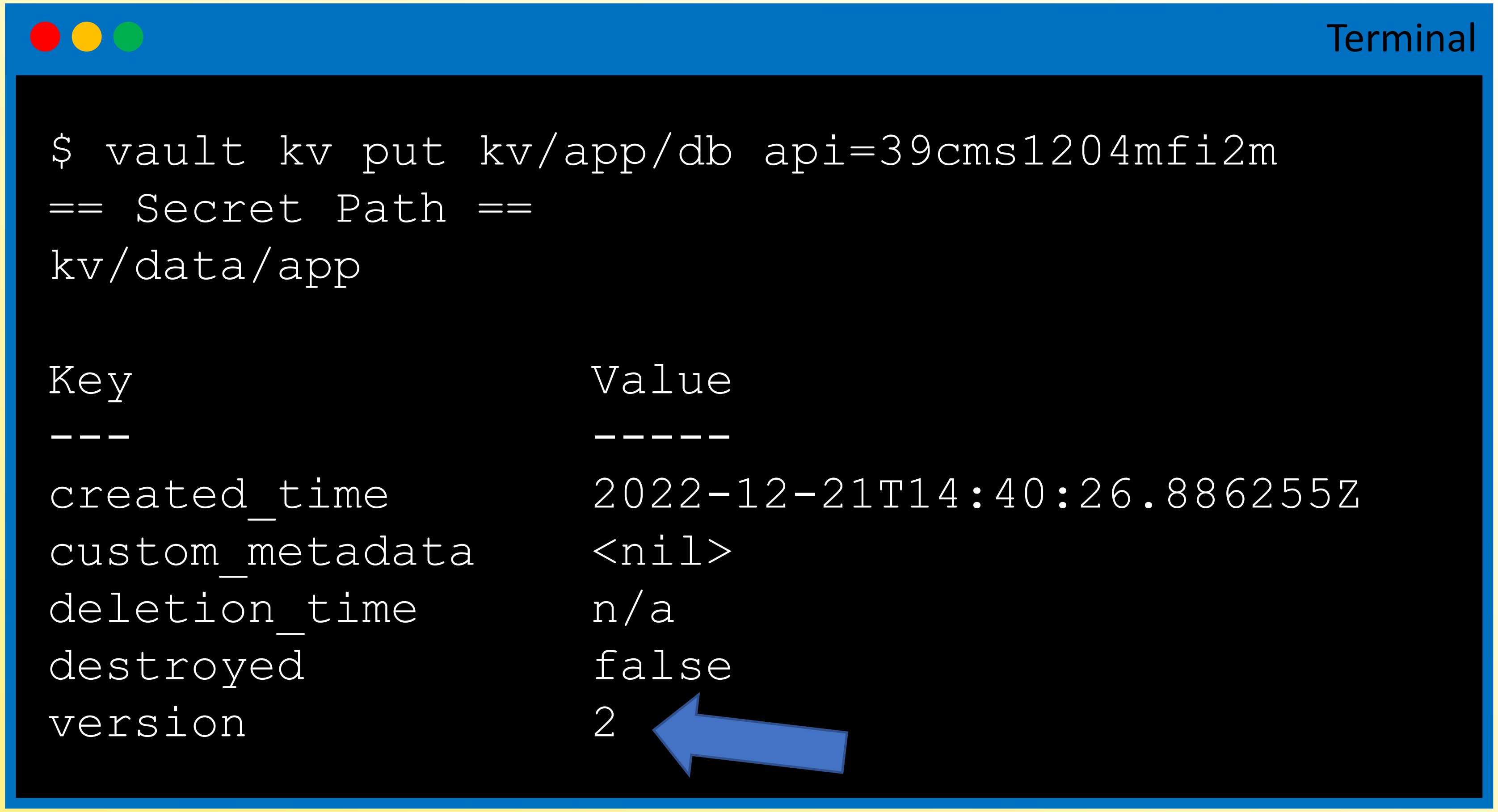
Writing Data to the KV Store

What will happen if I run the following command?


Terminal

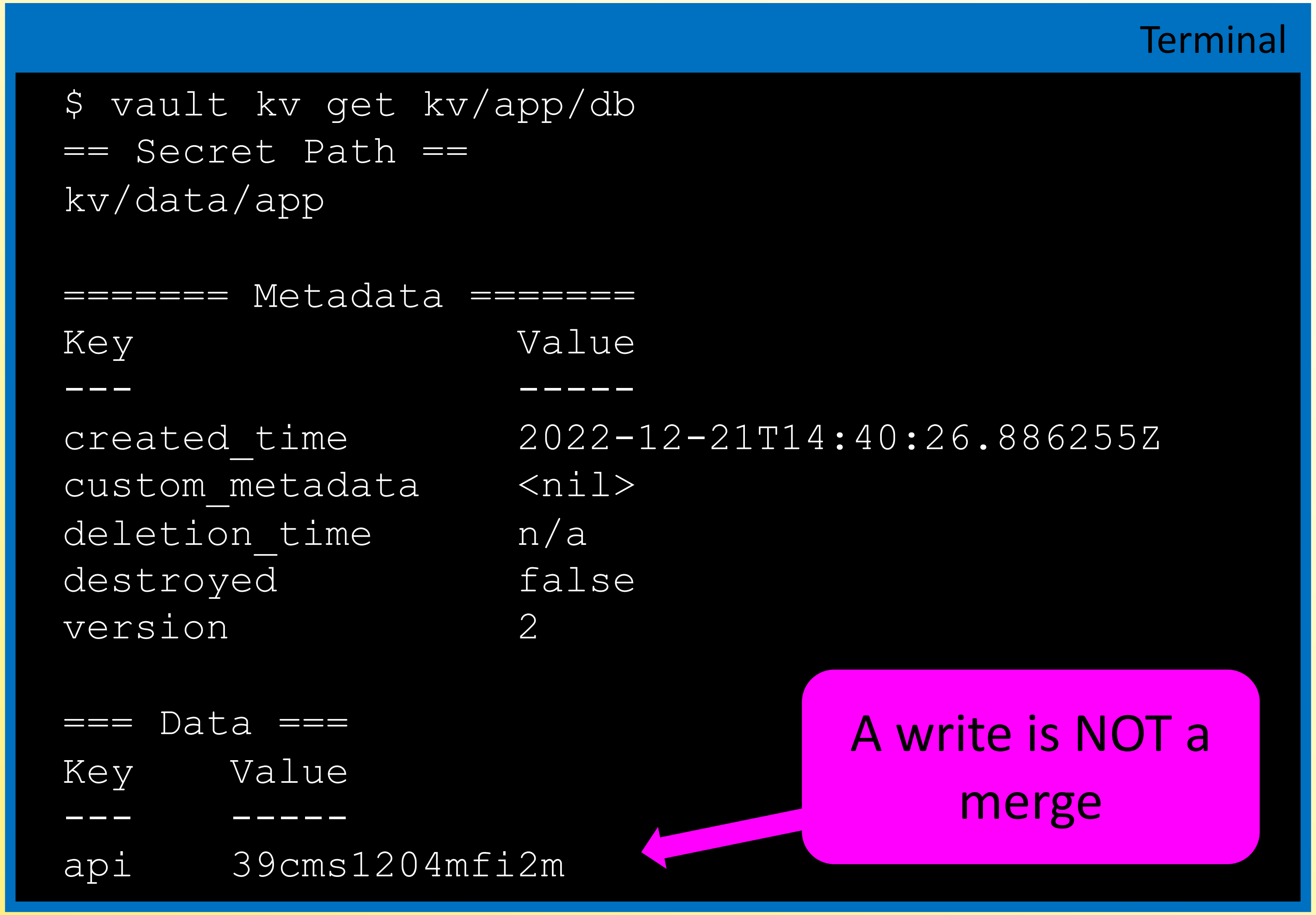
```
$ vault kv put kv/app/db api=39cms1204mfi2m
```

Writing Data to the KV Store



```
$ vault kv put kv/app/db api=39cms1204mfi2m
== Secret Path ==
kv/data/app

Key                  Value
---                  -----
created_time         2022-12-21T14:40:26.886255Z
custom_metadata      <nil>
deletion_time        n/a
destroyed            false
version              2
```

Writing Data to the KV Store


Terminal

```
$ vault kv get kv/app/db
== Secret Path ==
kv/data/app

======= Metadata =======
Key                Value
---                -----
created_time       2022-12-21T14:40:26.886255Z
custom_metadata    <nil>
deletion_time      n/a
destroyed          false
version            2

=== Data ===
Key    Value
---    -----
api    39cms1204mfi2m
```

A write is NOT a merge

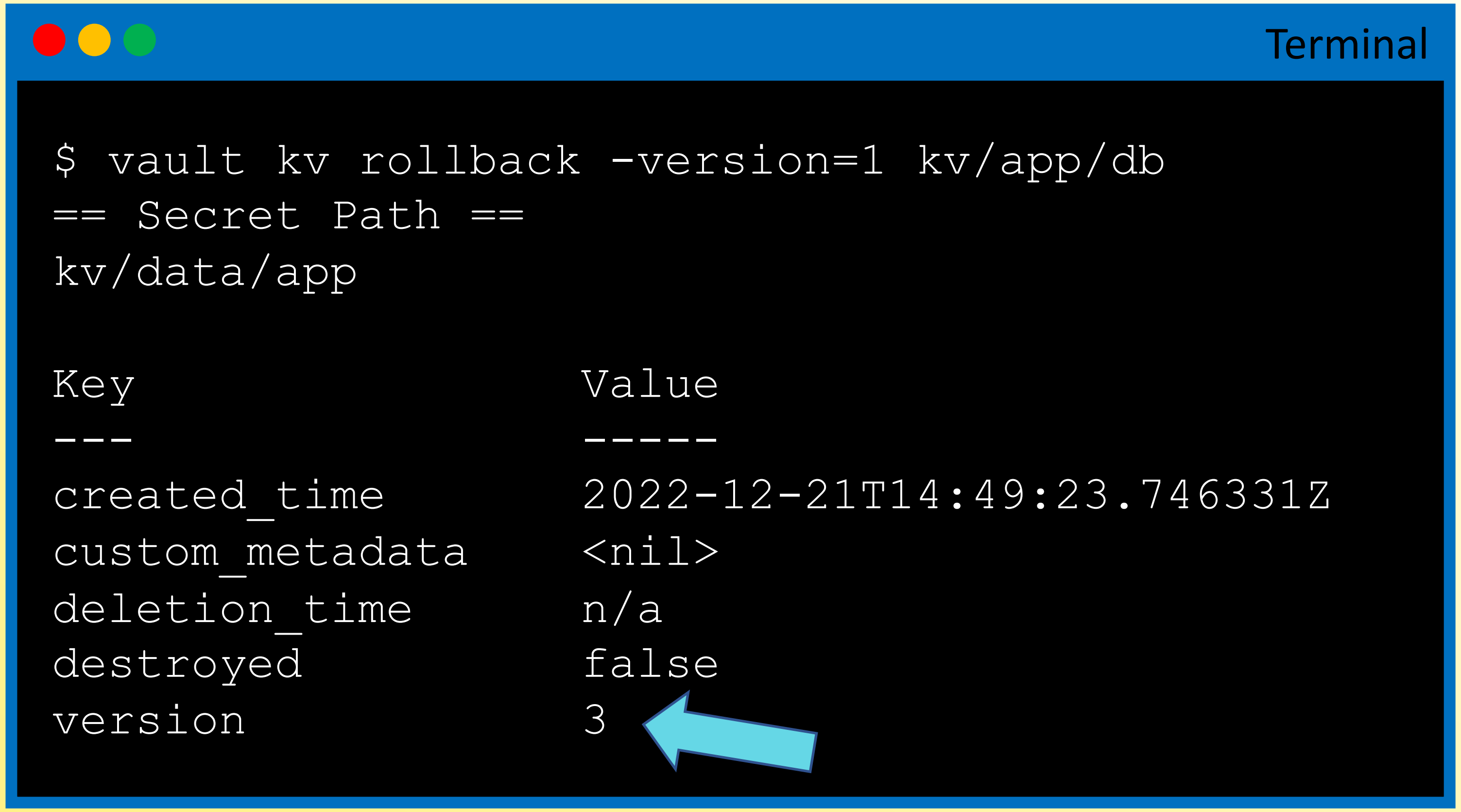
Writing Data to the KV Store

All you wanted to do was update the `api` but now you've lost your data?

What can we do?

Writing Data to the KV Store

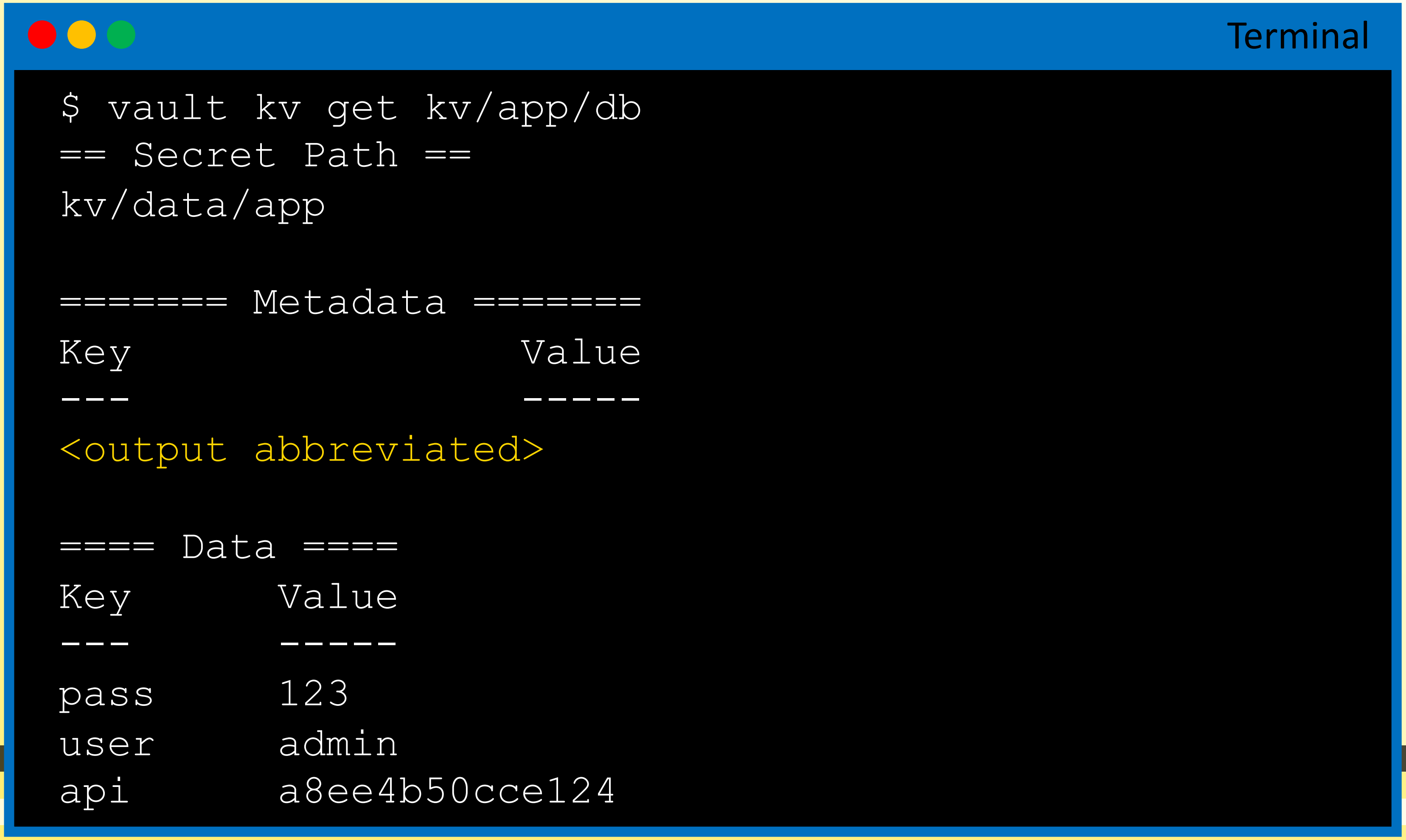
Recover Your Data using Rollback



```
$ vault kv rollback -version=1 kv/app/db
== Secret Path ==
kv/data/app

Key                  Value
---                  -----
created_time         2022-12-21T14:49:23.746331Z
custom_metadata      <nil>
deletion_time        n/a
destroyed            false
version              3
```

Writing Data to the KV Store

Recover Your Data Using Rollback

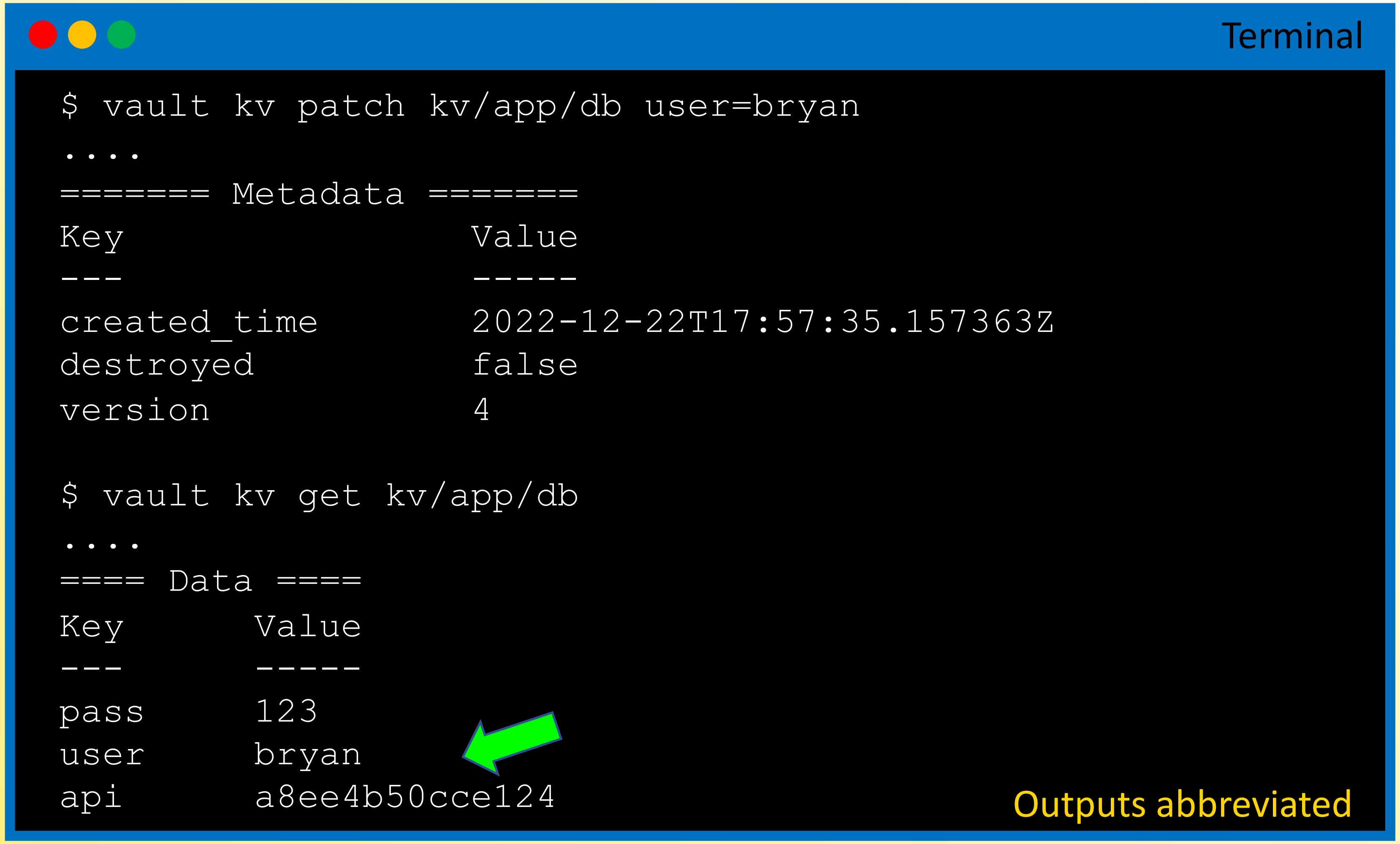


Terminal

```
$ vault kv get kv/app/db
== Secret Path ==
kv/data/app

======= Metadata =======
Key                  Value
---                  -----
<output abbreviated>

==== Data ====
Key        Value
---        -----
pass       123
user       admin
api        a8ee4b50cce124
```

Writing Data to the KV Store

Patch Your Data



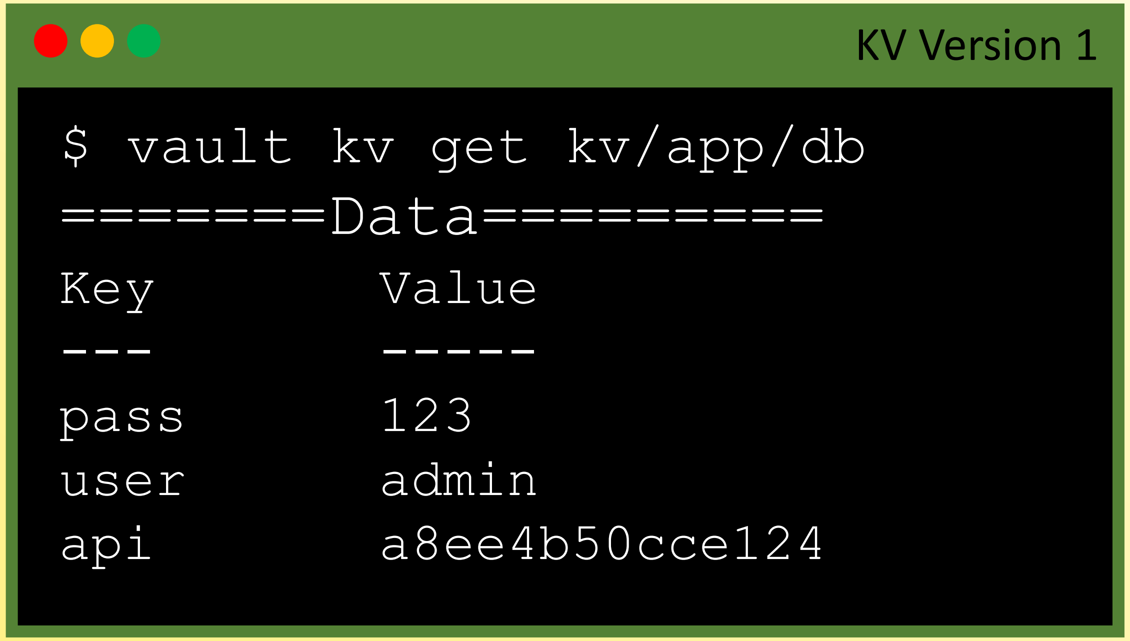
Terminal

```
$ vault kv patch kv/app/db user=bryan
....
======= Metadata =======
Key                    Value
---                    -----
created_time           2022-12-22T17:57:35.157363Z
destroyed              false
version                4

$ vault kv get kv/app/db
....
==== Data ====
Key        Value
---        -----
pass       123
user       bryan
api        a8ee4b50cce124
```

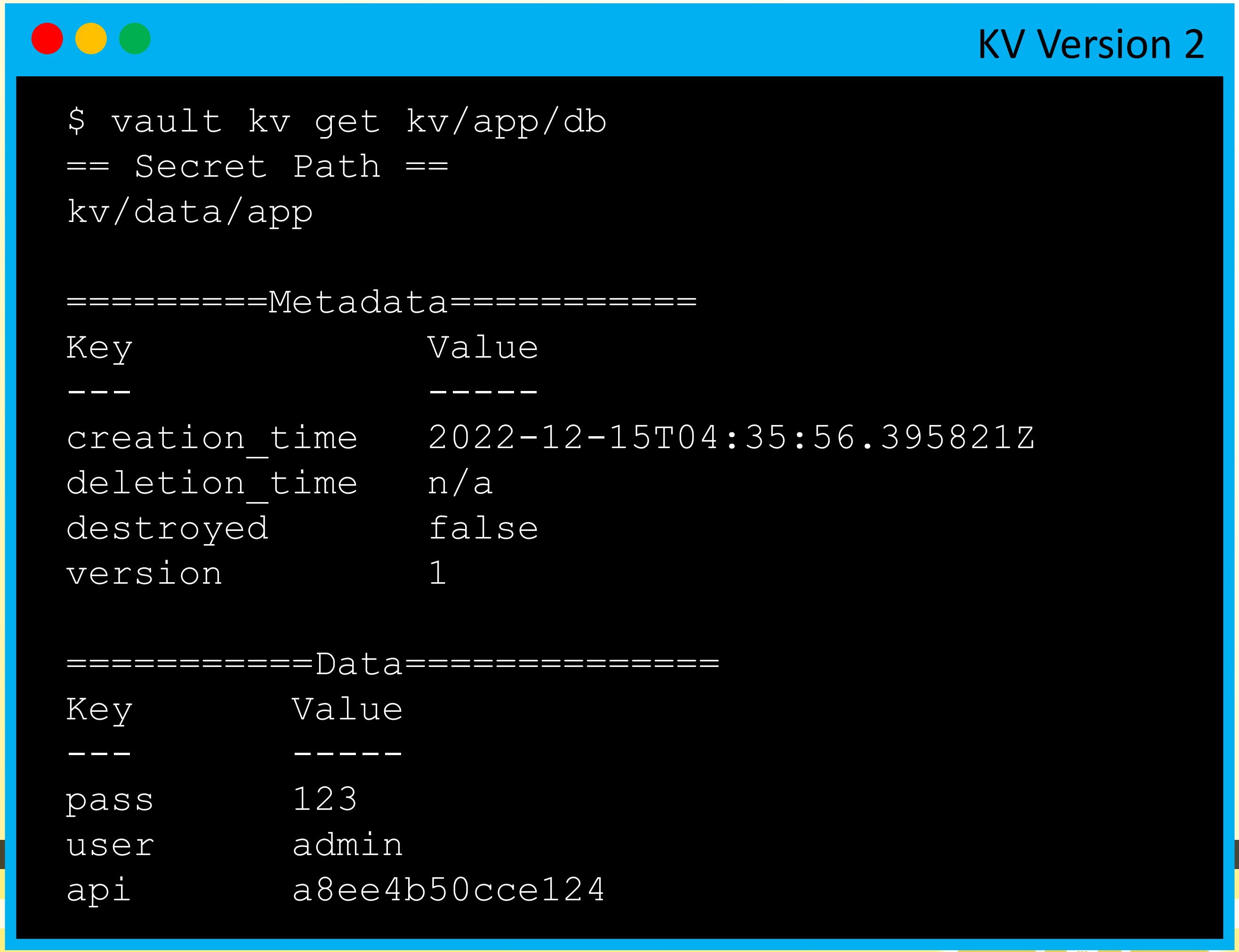
Outputs abbreviated

Reading Data from the KV Store

Compare Version 1 and Version 2


KV Version 1

```
$ vault kv get kv/app/db
=======Data==========
Key        Value
---        -----
pass       123
user       admin
api        a8ee4b50cce124
```

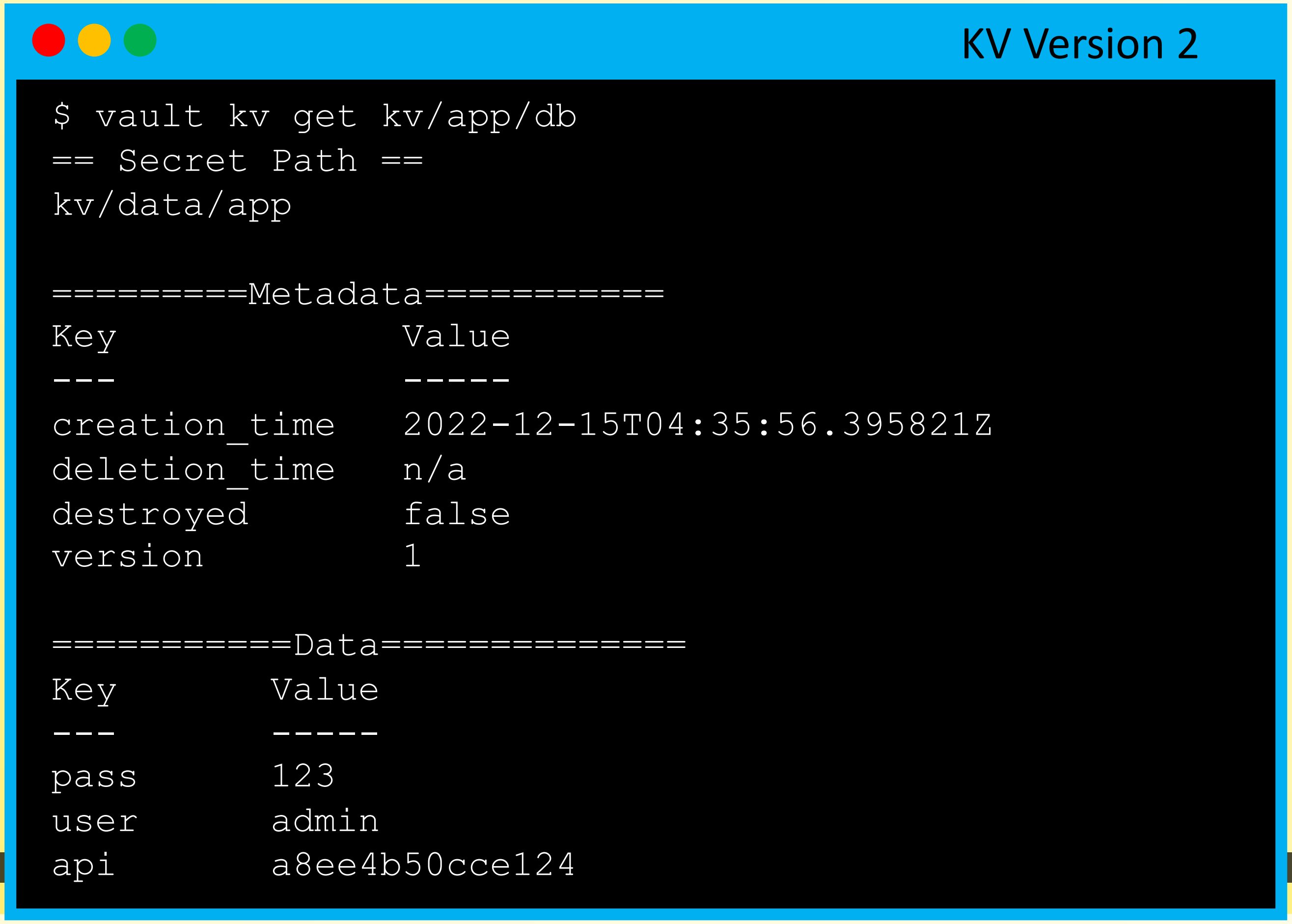

KV Version 2

```
$ vault kv get kv/app/db
== Secret Path ==
kv/data/app

=========Metadata===========
Key                Value
---                -----
creation_time      2022-12-15T04:35:56.395821Z
deletion_time      n/a
destroyed          false
version            1

===========Data===============
Key          Value
---          -----
pass         123
user         admin
api          a8ee4b50cce124
```

The CLI command is the same, but we get different output behavior

Reading Data from the KV Store

Output



KV Version 2

```
$ vault kv get kv/app/db
== Secret Path ==
kv/data/app

=========Metadata===========
Key                Value
---                -----
creation_time      2022-12-15T04:35:56.395821Z
deletion_time      n/a
destroyed          false
version            1

===========Data==============
Key          Value
---          -----
pass         123
user         admin
api          a8ee4b50cce124
```

Default output type is `table`

Reading Secrets from the KV Store

Output

KV Version 2

```
$ vault kv get -format=json kv/app/db
{
  "request_id": "249fca06-a8ce-5617-d598-1c12384d4ac8",
  "lease_id": "",
  "lease_duration": 0,
  "renewable": false,
  "data": {
    "data": {
      "pass": "123",
      "user": "admin",
      "api": " a8ee4b50cce124",
    },
    "metadata": {
      "created_time": "2022-12-21T13:59:29.917893Z",
      "custom_metadata": null,
      "deletion_time": "",
      "destroyed": false,
      "version": 1
```

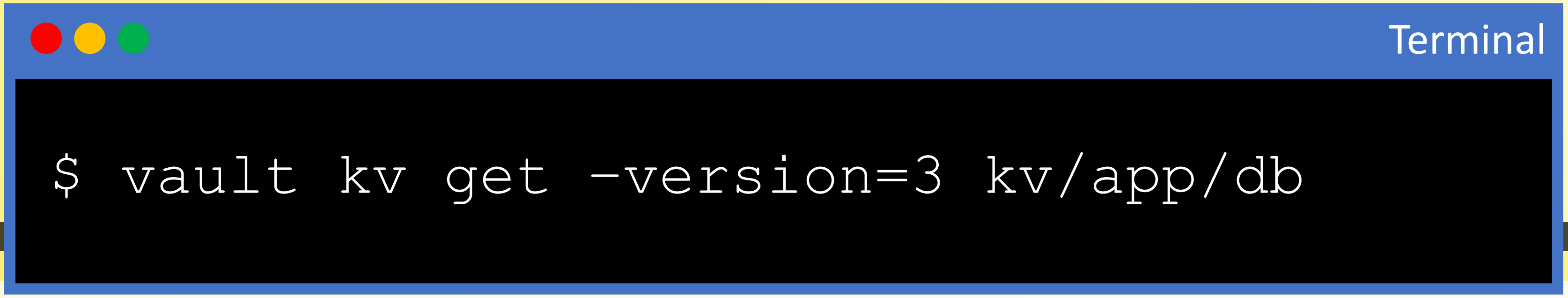
Output abbreviated

Change output format to `json`

Reading Secrets from the KV Store

Important Things to Remember

- A regular `read` request will return the latest version of the secret
- If the latest version of the secret has been deleted (KV V2), it will return the related metadata but no data (secrets)
- You can read a previous version of a secret (if one exists) by adding the `-version=x` flag to the request



```
$ vault kv get -version=3 kv/app/db
```

Deleting Secrets from the KV Store

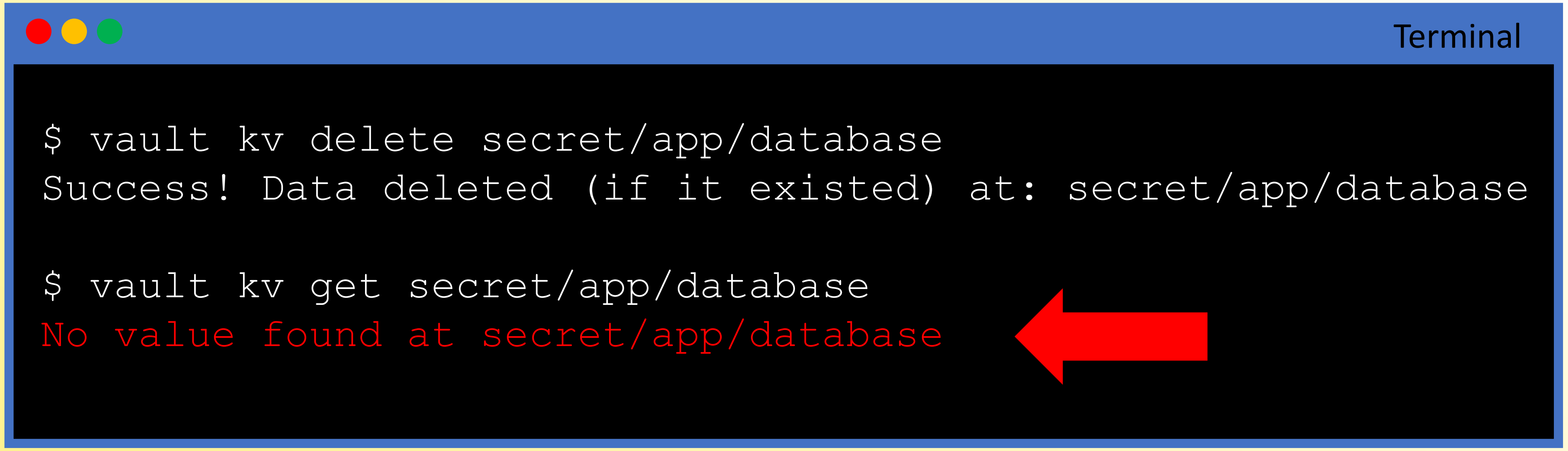
- A `delete` on KV V1 is a delete – the data is destroyed
    - You'd have to restore Vault/Consul to retrieve the old data

- A `delete` on KV V2 is a soft delete – data is not destroyed
    - Data can be restored with a `undelete/rollback` action

- A `destroy` (only KV V2) is a permanent action – destroyed on disk
    - Cannot be restored except for a Vault/Consul restore action

Deleting Secrets in the KV Store

KV Version 1 – Read Output after Deleting

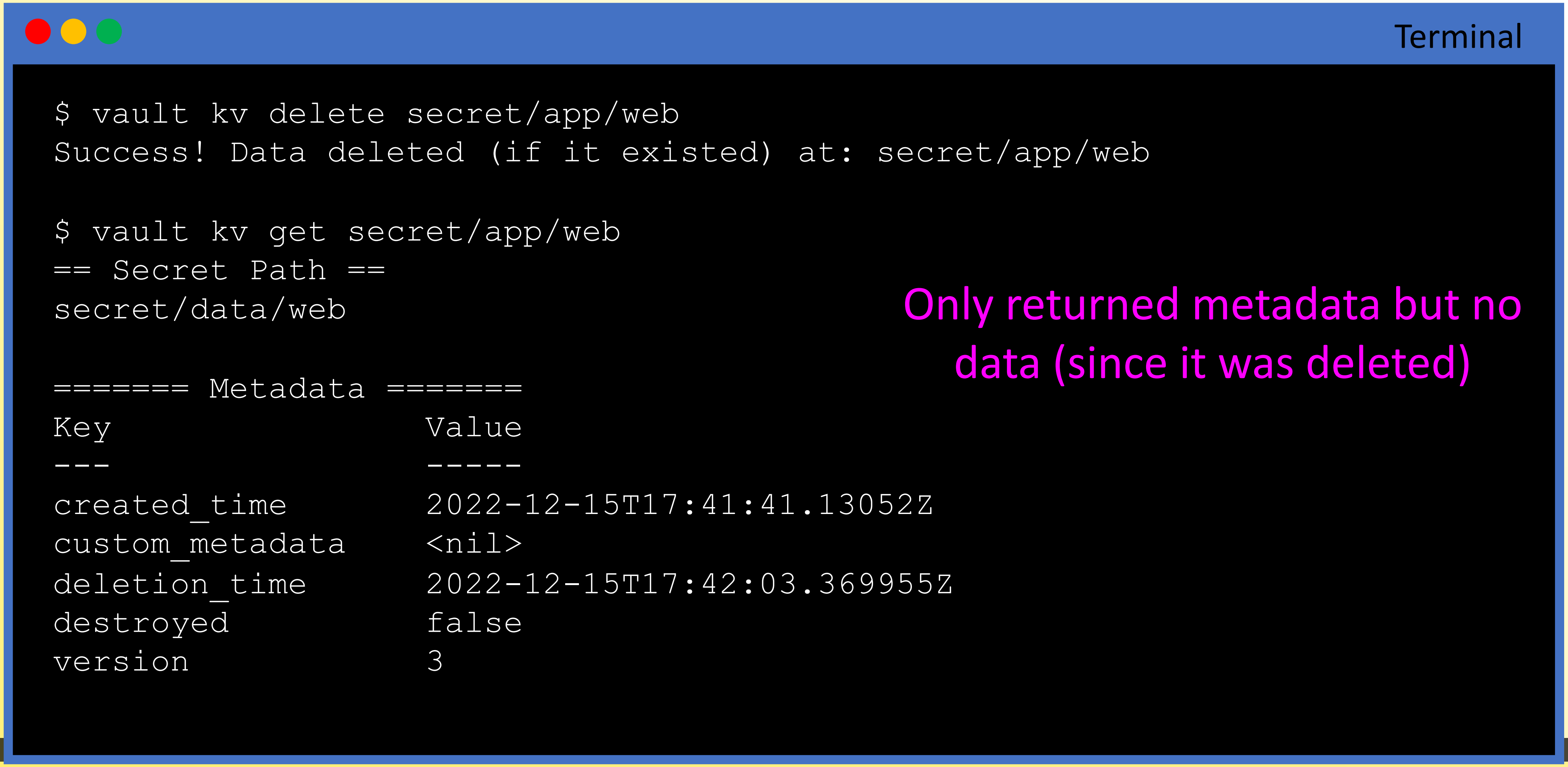


```
$ vault kv delete secret/app/database
Success! Data deleted (if it existed) at: secret/app/database

$ vault kv get secret/app/database
No value found at secret/app/database
```

No values exist if you delete a secret at a path for KV V1

Deleting Secrets from the KV Store

KV Version 2 – Read Output after Deleting the Latest Version



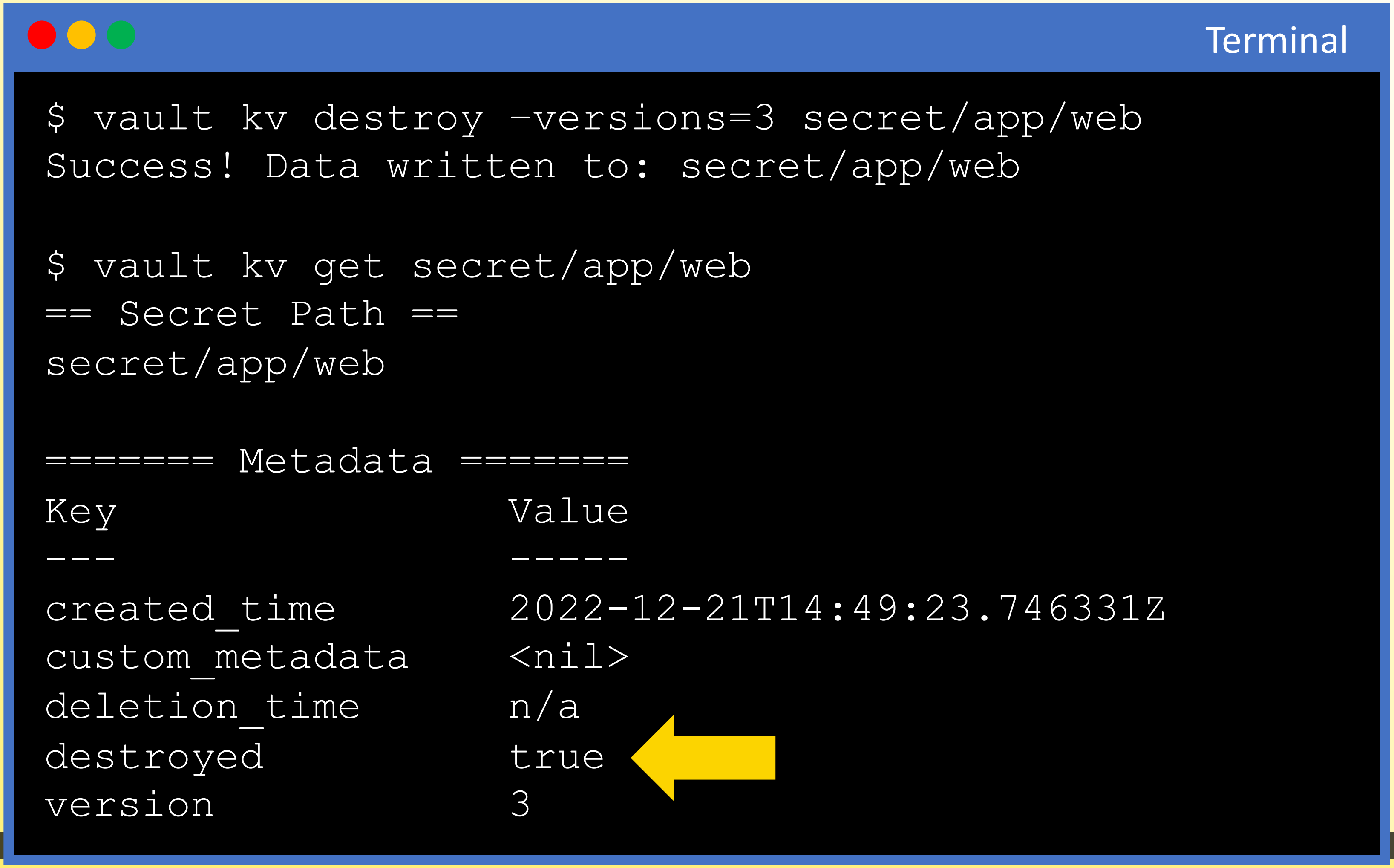
Terminal

```
$ vault kv delete secret/app/web
Success! Data deleted (if it existed) at: secret/app/web

$ vault kv get secret/app/web
== Secret Path ==
secret/data/web

======= Metadata =======
Key                Value
---                -----
created_time       2022-12-15T17:41:41.13052Z
custom_metadata    <nil>
deletion_time      2022-12-15T17:42:03.369955Z
destroyed          false
version            3
```

Only returned metadata but no data (since it was deleted)

Destroy Secrets from the KV Store

KV Version 2 – Read Output after Destroying the Latest Version



```
$ vault kv destroy -versions=3 secret/app/web
Success! Data written to: secret/app/web

$ vault kv get secret/app/web
== Secret Path ==
secret/app/web

======= Metadata =======
Key                  Value
---                  -----
created_time         2022-12-21T14:49:23.746331Z
custom_metadata      <nil>
deletion_time        n/a
destroyed            true
version              3
```

Database Secrets Engine

Intro to Database Secrets Engine

- The database secrets engine generates dynamic credentials against one or more databases – username and password

- Generated credentials are based on roles which provides a one-to-one mapping to a permission set on the targeted database

- Credentials are tied to a lease – credentials are revoked when the lease expires
  - Vault reaches back out to the database and deletes the credentials (no technical debt)

Database Secrets Engine - Plugins

Vault
CERTIFIED
OPERATIONS
PROFESSIONAL

Database Secrets Engine

- Cassandra
- InfluxDB
- Oracle
- Couchbase
- MongoDB
- PostgreSQL
- Elastisearch
- MongoDB Atlas
- Redshift
- HanaDB
- MSSQL
- Snowflake
- IBM Db2
- MySQL/MariaDB
- Custom

Database Secrets Engine - Configuration

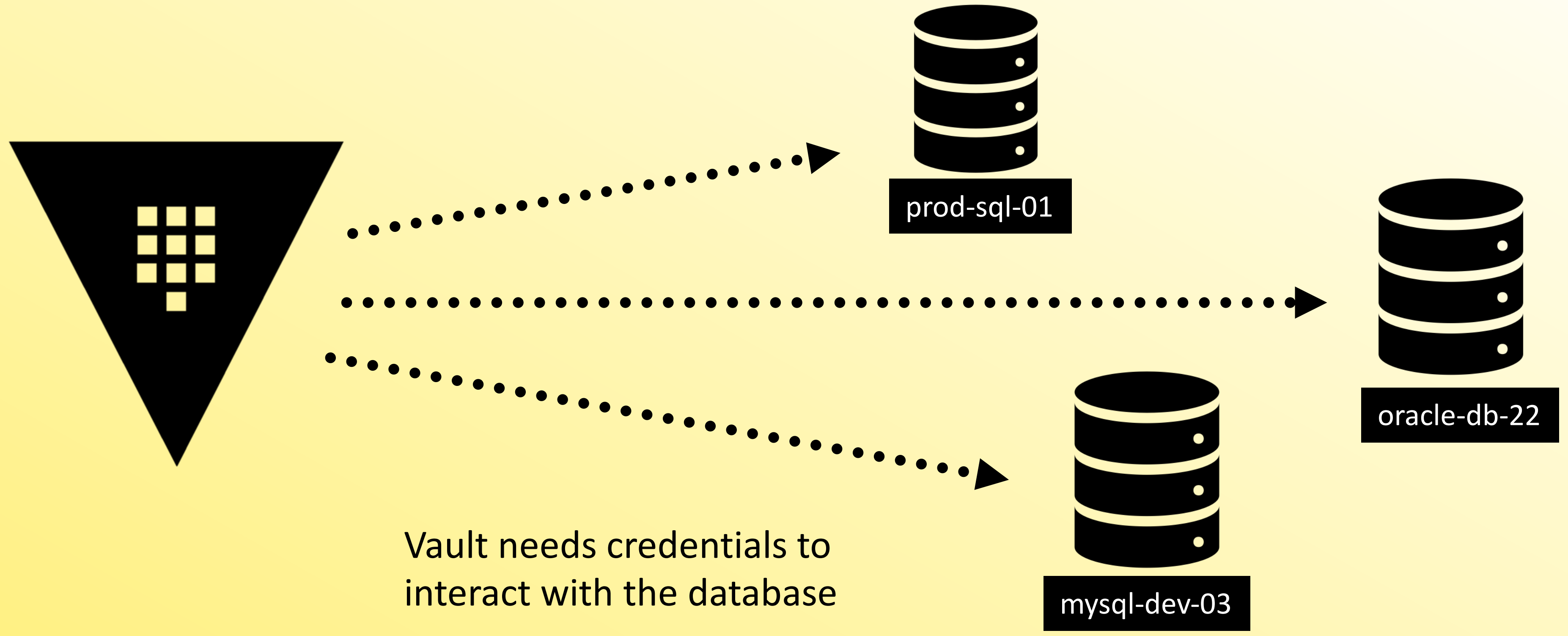
There are generally two steps when configuring a secrets engine that will generate dynamic credentials:

1 Configure Vault with access to the database

2 Configure roles based on permissions required

Database Secrets Engine – DB Config

Configure Vault with access to the database



Vault needs credentials to interact with the database

Database Secrets Engine – DB Config

Vault
CERTIFIED
OPERATIONS
PROFESSIONAL

Provide credentials to a secrets engine that gives Vault permission to create, list, and delete credentials on the platform:

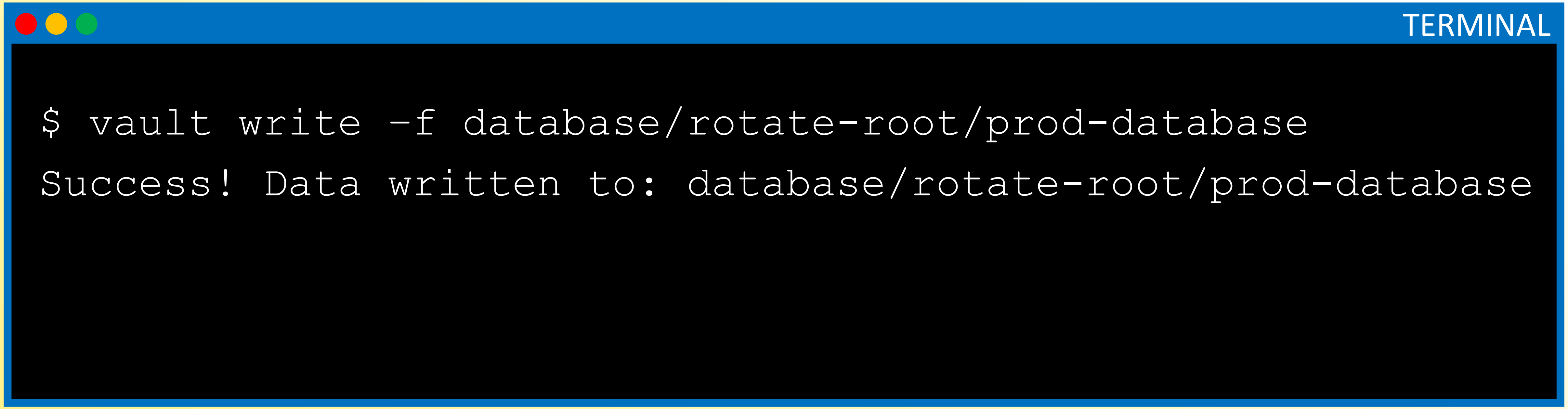
TERMINAL

```
$ vault write database/config/prod-database \
    plugin_name=mysql-database-plugin \
    connection_url="{{username}}:{{password}}@tcp(prod.hcvop.com:3306)/" \
    allowed_roles="app-integration, app-hcvop" \
    username="vault-admin" \
    password="vneJ4908fkd3084Bmrk39fmslslf#e&349"
```

Rotate Root Credentials

- You can easily rotate the root credentials for each database configuration using the rotate-root endpoint
- This allows you to meet any internal policies where credentials should be frequently rotated
- Also ensures that only Vault and the Database server know the root credentials – *no human would know the creds*.

Rotate Root Credentials



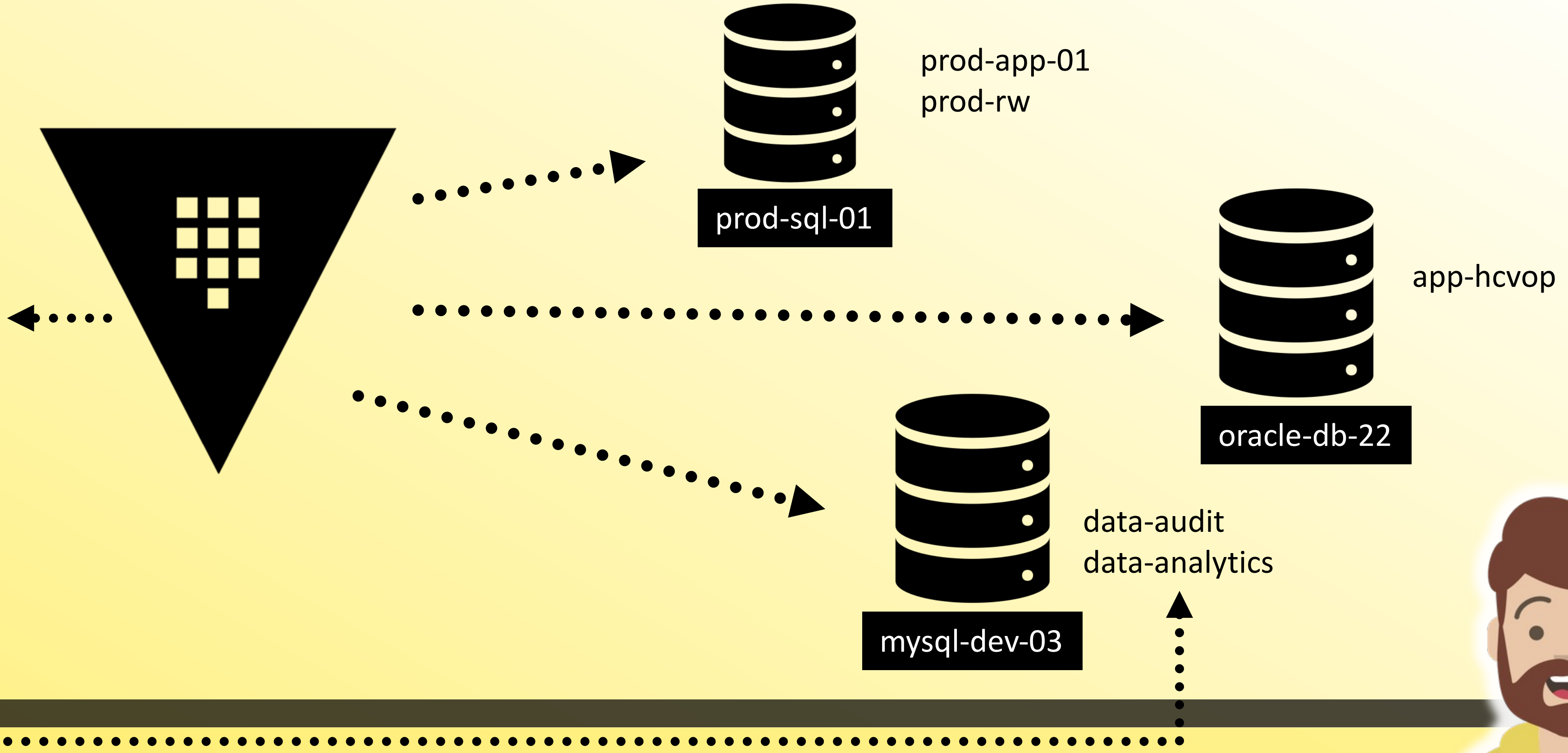
```
$ vault write -f database/rotate-root/prod-database
Success! Data written to: database/rotate-root/prod-database
```

Database Secrets Engine - Roles

Configure roles based on permissions required

Roles

- prod-app-01
- prod-rw
- app-hcvop
- data-audit
- data-analytics



Database Secrets Engine – Create a Role

```
$ vault write database/roles/app-hcvop \
    db_name=prod_database \
    creation_statements="CREATE USER '{{name}}'@'%' IDENTIFIED
     BY '{{password}}';GRANT SELECT ON *.* TO '{{name}}'@'%';" \
    default_ttl="1h" \
    max_ttl="24h"
```

Database Secrets Engine – Create a Role

Breaking Down the Command

Vault
CERTIFIED
OPERATIONS
PROFESSIONAL

$$\texttt{vault}\ \underbrace{\texttt{write}}_{\text{Use write when creating a new role}}\ \underbrace{\texttt{database}}_{\text{Database secrets engine mount}}\texttt{/}\underbrace{\texttt{roles}}_{\text{Path where roles are created/stored}}\texttt{/}\underbrace{\texttt{app-hcvop}}_{\text{Name of the role to be created}}$$

Database Secrets Engine – Create a Role

Important Parameters for a Role

```
db_name="prod_database"
```

Maps to the database config that you want to generate credentials on

```
creation_statements="CREATE USER '{{name.."
```

The statement to be executed on the database to create the user with the appropriate permissions

```
default_ttl="1h"
```

Initial Lease time for the generated dynamic credential

```
max_ttl="24h"
```

Make the lease renewable up to a certain timeframe

Static Roles

- Static role is a 1-1 mapping of a role to an existing, static user on the database.
- Vault will rotate the credential based on a configurable time period but the user does not change
- *Example Use Case:* Use this if an application requires a specific username to connect to the database but the password can change.

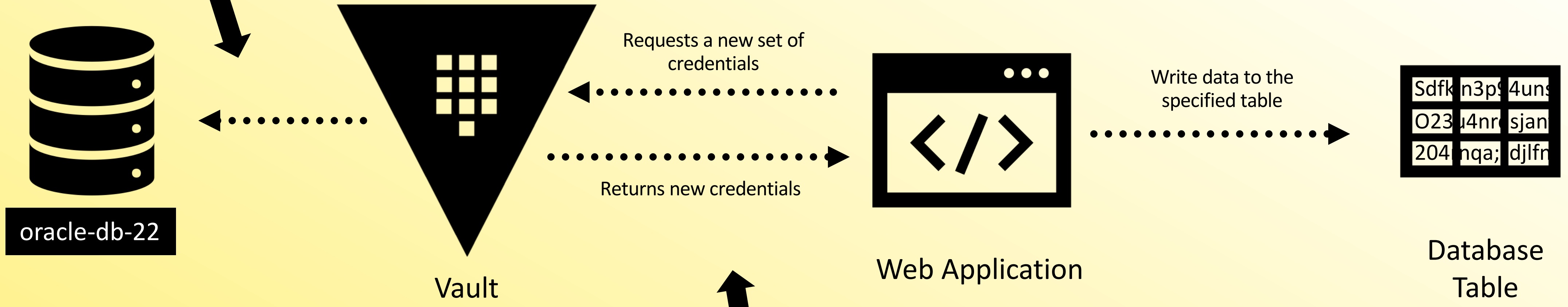
Password Policies

- Can set password policies per database configuration, not per role
- Each database has a default password policy defined as: 20 characters with at least 1 uppercase character, at least 1 lowercase character, at least 1 number, and at least 1 dash character.
- Note that some of the credentials generated by different plugins might be different due to the supported platform

Database Secrets Engine – Generate Creds

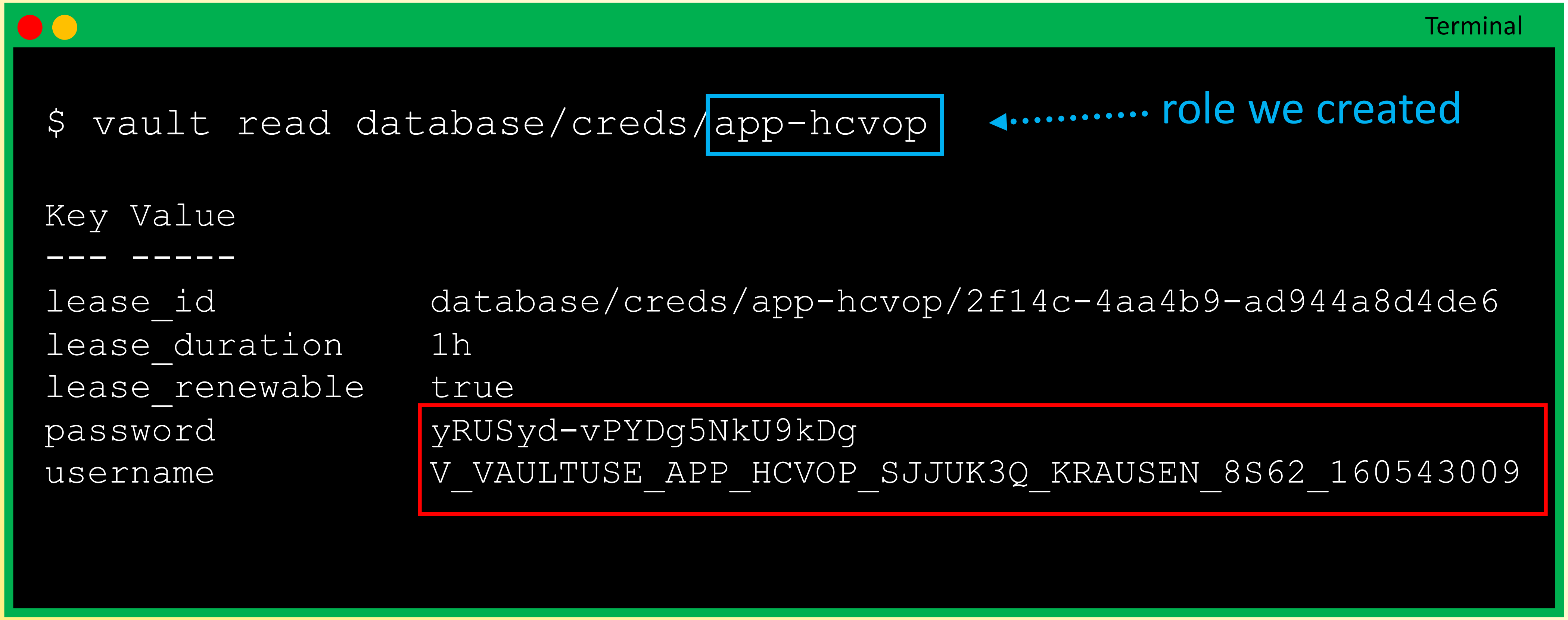
Vault uses the credentials we provided the database secrets engine to generate the dynamic credentials for the requested role

Vault client references the Vault role when requesting dynamic credentials



Credentials returned are attached to a lease (TTL) and will expire after that TTL unless renewed by the application

Database Secrets Engine – Generate Creds



```
$ vault read database/creds/app-hcvop

Key               Value
---               -----
lease_id          database/creds/app-hcvop/2f14c-4aa4b9-ad944a8d4de6
lease_duration    1h
lease_renewable   true
password          yRUSyd-vPYDg5NkU9kDg
username          V_VAULTUSE_APP_HCVOP_SJJUK3Q_KRAUSEN_8S62_160543009
```

Permissions for Generating Database Creds

- Like anything else in Vault, a Vault client requires a token with a policy that has permission to generate database credentials

```
# Get credentials from the database secrets engine

path "database/creds/app-hcvop" {
  capabilities = ["read"]
}
```