Vault

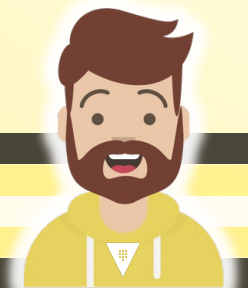
CERTIFIED
OPERATIONS
PROFESSIONAL

Auth Methods

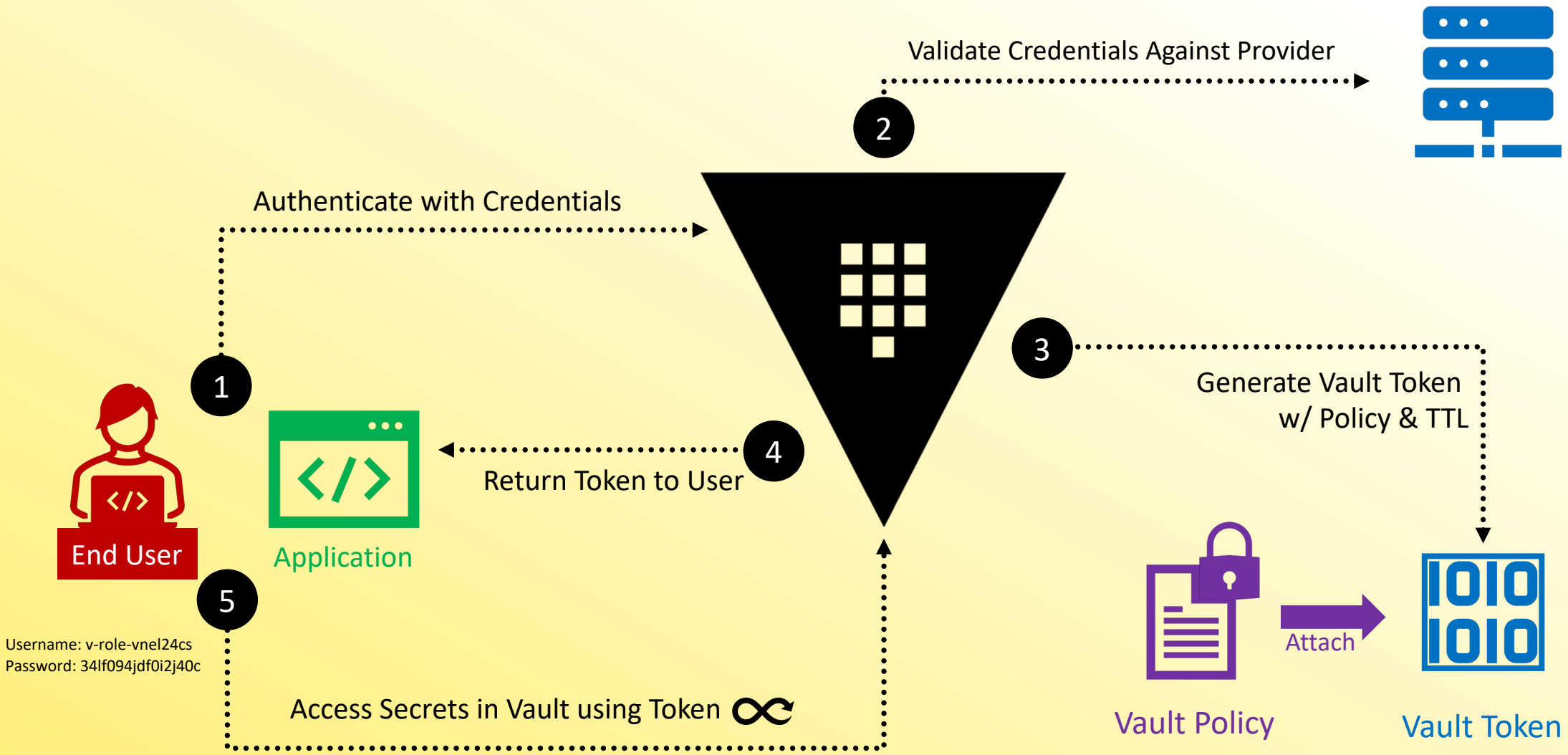
Introduction to Auth Methods

- Vault components that perform authentication and manage identities
- Responsible for assigning identity and policies to a user
- Multiple authentication methods can be enabled depending on your use case
  - Auth methods can be differentiated by human vs. system methods

- Once authenticated, Vault will issue a client token used to make all subsequent Vault requests (read/write)
  - The fundamental goal of all auth methods is to obtain a token
  - Each token has an associated policy (or policies) and a TTL



Auth Methods Workflow

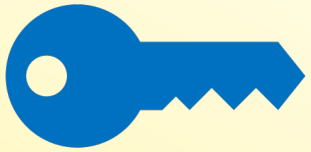


Auth Methods

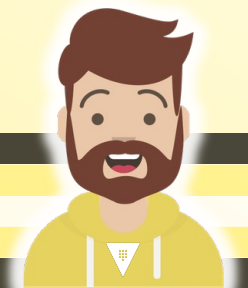
Auth Methods Likely on the Exam

AppRole

Tokens

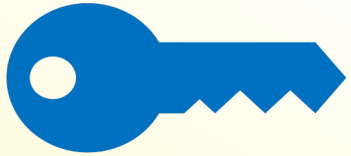


UserPass



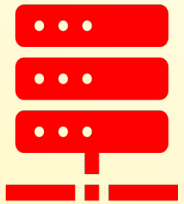
Auth Methods

Human-Based



Userpass

- Integrates with an Existing Identity Provider
- Requires a Hands-On Approach to Use
- Logging in via Prompt or Pop-up
- Often configured with the Platforms Integrated MFA



RADIUS

GitHub

JWT/OIDC

okta

Auth Methods

System-Based

- Uses non-human friendly methodologies (not easy to remember)
- Usually Integrates with an Existing Platform
- Vault validates credentials with the platform

aws

TOKENS

Vault CERTIFIED OPERATIONS PROFESSIONAL

CLOUDFOUNDRY

TLS Certificates

Kerberos

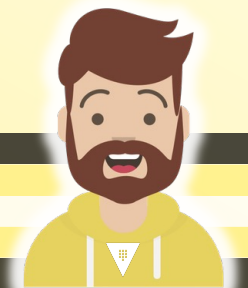
ORACLE CLOUD

AppRole

Microsoft Azure

Alibaba Cloud

kubernetes



Working with Auth Methods

- Most auth methods <u>must be enabled</u> before you can use them
- One or many auth methods can be used at any given time
  - Generally different auth methods are used for different use cases (app vs. human)

- The token auth method is enabled by default, and you cannot enable another nor disable the tokens auth method
  - New Vault deployment will use a token for authentication
  - <u>Only</u> method of authentication for a new Vault deployment is a root token

Working with Auth Methods

Auth methods can be enabled/disabled and configured using the UI, API, or the CLI

- Note that the UI isn't fully-featured like the CLI and API, so there might be things you can't do in the UI

You must provide a valid token to enable, disable, or modify auth methods in Vault. The token must also have the proper privileges.

Working with Auth Methods

Vault CERTIFIED OPERATIONS PROFESSIONAL

Use the `vault auth` command

- `enable`
- `disable`
- `list`
- `tune`
- `help`

Terminal

```
$ vault auth enable approle
Success! Enabled approle auth method at: approle/
```

Terminal

```
$ vault auth disable approle
Success! Disabled the auth method (if it existed) at: approle/
```

Terminal

```
$ vault auth list
Path              Type       Accessor                 Description
----              ----       --------                 -----------
hcvop/            approle    auth_approle_d8c20abe    n/a
token/            token      auth_token_89ce3371      token based credentials
vault-course/     approle    auth_approle_b3f0c92d    n/a
```

Working with Auth Methods

Enable an Auth Method at the Default Path

Terminal

```
$ vault auth enable approle
Success! Enabled approle auth method at: approle/
```

Enable an Auth Method using a Custom Path

Terminal

```
$ vault auth enable -path=hcvop approle
Success! Enabled approle auth method at: hcvop/
```

Working with Auth Methods

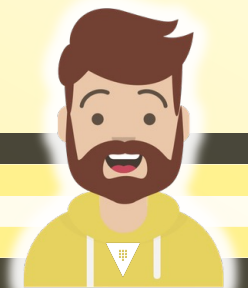
Enable Auth Method Using Default Path via CLI

```
$ vault auth enable approle
```

Type of Vault object you want to work with

Subcommand

Type of Auth Method

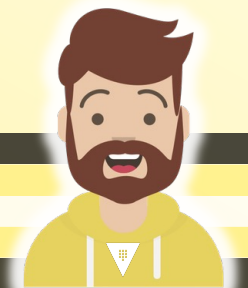


Working with Auth Methods

Enable Auth Method with Custom Path via CLI

```
$ vault auth enable -path=apps approle
```

- `auth` — Type of Vault object you want to work with
- `enable` — Subcommand
- `-path=apps` — Customize the Path Name
- `approle` — Type of Auth Method



Working with Auth Methods

After the auth method has been enabled, use the auth prefix to configure the auth method:

- Syntax: `vault write auth/<path name>/<option>`

Terminal

```
$ vault write auth/approle/role/hcvop \
    secret_id_ttl=10m \
    token_num_uses=10 \
    token_ttl=20m \
    token_max_ttl=30m \
    secret_id_num_uses=40
```

Working with Auth Methods

Using the API

Enabling an Auth Method:

- Method: POST

Don't forget you need a valid token

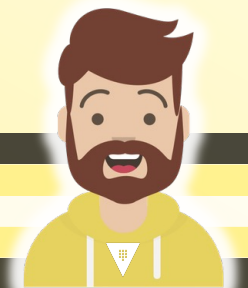
Alternatively, you can point to a file here if you want `--data @data.json`

Enable Auth Method

Terminal

```
$ curl \
    --header "X-Vault-Token: s.v2otcpHygZHWiD7BQ7P5aJjL" \
    --request POST \
    --data '{"type": "approle"}' \
    https://vault.ncvop.com:8200/v1/sys/auth/approle
```

API Endpoint



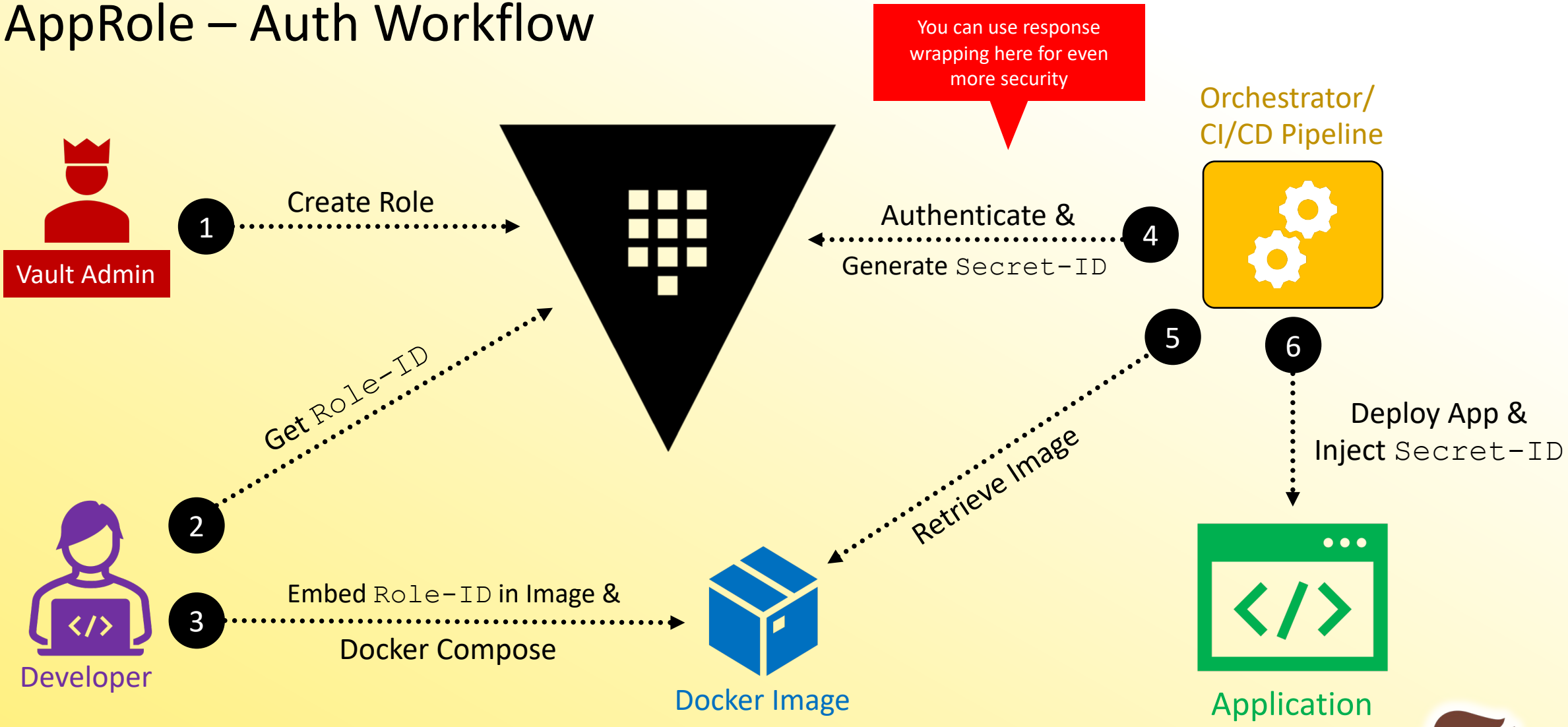
AppRole Auth Method

Introduction to AppRole

- AppRole auth method enables machines or applications to authenticate to Vault using a pre-defined *role*.
  - A *role* represents a one-to-one mapping between client authentication and the Vault permission requirements
  - Each defined role has a static *role-id* and can have zero or many *secret-id*s that can be generated and used for authentication
    - *Example: A fleet of web servers can all use the same role-id but each have a unique secret-id*

- This auth method is oriented for use by machines/automated services and is not very useful for human clients

AppRole – Auth Workflow



AppRole – Configuration Workflow

Needed for authentication

Vault with
AppRole Auth Method

Create Role

AppRole Role

Read Role-ID

Generate Secret-ID

Role-ID

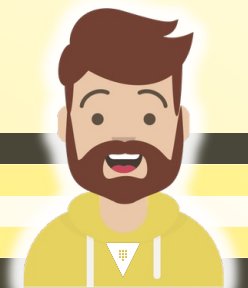
22549d0d-147a-d6e2-fa2e-9cedd3b20977

Role-ID will always be the same for each unique role you create

b30f2778-9943-f930-1683-2d31973e285f

Secret-ID

Secret-ID will be unique each time you generate one



AppRole – Configuration Workflow

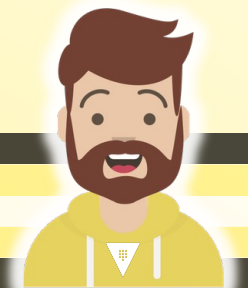
Needed for authentication

Fleet of Web Severs

Requires Identical Permissions in Vault

	Application	Application	Application
Role-ID	22549d0d-147a-d6e2-fa2e-9cedd3b20977	22549d0d-147a-d6e2-fa2e-9cedd3b20977	22549d0d-147a-d6e2-fa2e-9cedd3b20977
Secret-ID	b30f2778-9943-f930-1683-2d31973e285f	0514b3b1-e1ce-2741-0b57-ef836c29c7d3	d2628ca1-4683-a285-f55d-186d9b10e530

Unique Secret-ID for each Workload



Enable AppRole Auth Method

```
# Enable AppRole at the default path
$ vault auth enable approle
Success! Enabled approle auth method at: approle/


# Enable AppRole at a custom path
$ vault auth enable -path=hcvop approle
Success! Enabled approle auth method at: hcvop/
```

Create a New Role

```
# Create a new role named 'hcvop'

$ vault write auth/approle/role/hcvop \
    token_policies=web-app \
    token_ttl=1h \
    token_max_ttl=24h
Success! Data written to: auth/approle/role/hcvop/
```

Create a New Role

Breaking Down the CLI Command


```
vault write auth/approle/role/hcvop
```

- `write` — Subcommand used to read, write, delete, or list a role
- `auth` — Default path for auth methods (not configurable)
- `approle` — Path where the AppRole auth method was enabled
- `role` — Path where roles are created and stored (not configurable)
- `hcvop` — The name of the role you want to create, modify, or delete

Create a New Role

A Few Configuration Tips for AppRole

- You can set the TTL of the resulting token
  - `token_max_ttl=24h`
  - `token_ttl=1h`
- You can set the TTL of the secret-id that you have generated
  - `secret_id_ttl=24h`
- You can configure CIDR restrictions for a role (binds the resulting token as well)
  - `token_bound_cidrs="10.1.16.0/16"`
- You can change the resulting token type to a batch token
  - `token_type=batch`

These are configuration parameters for an AppRole role

Reading the Role-ID

```
# Read the role-id for a particular role

$ vault read auth/approle/role/hcvop/role-id
Key         Value
---         -----
role_id     22549d0d-147a-d6e2-fa2e-9cedd3b20977
```

Every time you run this command for the **same role name**, you will get the same exact value in response

Read a Role Configuration

```
# Read the current configuration of the role named 'hcvop'

$ vault read auth/approle/role/hcvop
Key                        Value
---                        -----
bind_secret_id             true
local_secret_ids           false
policies                   [web-app]
secret_id_bound_cidrs      <nil>
secret_id_num_uses         0
secret_id_ttl              0s
token_bound_cidrs          [10.1.16.0/16]
token_explicit_max_ttl     0s
token_max_ttl              24h
token_no_default_policy    false
token_num_uses             0
token_period               0s
token_policies             [web-app]
token_ttl                  1h
token_type                 default
```

Generating a Secret-ID

```
# Generate a secret-id for a particular role

$ vault write -f auth/approle/role/hcvop/secret-id
Key                   Value
---                   -----
secret_id             0514b3b1-e1ce-2741-0b57-ef836c29c7d3
secret_id_accessor    da025e1f-7247-1888-218c-37382d31e98e
secret_id_ttl         24h
```

Every time you run this command for the **same role name**, you will get a **different** secret-id

You will need to include `-f` or `-force` to run this command

Authenticate to Vault using AppRole

Via Command Line (CLI)

```
$ vault write auth/approle/login \
role_id=22549d0d-147a-d6e2-fa2e-9cedd3b20977 \
secret_id=b30f2778-9943-f930-1683-2d31973e285f

Key                     Value
---                     -----
token                   hvs.CAESIGjTXNYnz9T4h5y2_5Rt_d5ZNhW3dfgU4gQ
token_accessor          KmRLXSRbOzhXoA746g8g0KJu
token_duration          24h
token_renewable         true
token_policies          ["default" "web-app"]
identity_policies       []
policies                ["default" "web-app"]
token_meta_role_name    hcvop
```

Authenticate to Vault using AppRole

Via API

```
# Format the response using jq
$ curl \
    --request POST \
    --data '{"role_id":"22549d0d-...","secret_id":"b30f2778-..."}' \
    https://vault.hcvop.com:8200/v1/auth/approle/login | jq


# Authenticate and query for the client_token using jq
$ curl \
    --request POST \
    --data '{"role_id":"22549d0d-...","secret_id":"b30f2778-..."}' \
    https://vault.hcvop.com:8200/v1/auth/approle/login | jq -r
    '.auth.client_token'
```

Authenticate to Vault using AppRole

Via API


```
{
  "request_id": "c3750ef1-9fdf-b2db-0e7b-cfb433bf4120",
  "lease_id": "",
  "renewable": false,
  "lease_duration": 0,
  "data": null,
  "wrap_info": null,
  "warnings": null,
  "auth": {
    "client_token": "hvs.CAESIIJCoQiCpci-U0xiqGr0fG9pP0SHDgfVB96cti7bSI3aGiEKHGh2cy5vdWlPd2pEZ2RwVHZaWm95VWZoMnN1bmkQywE",
    "accessor": "rlbbs2nFlRvr348p9qaALM7O",
    "policies": [
      "default",
      "web-app"
    ],
    "token_policies": [
      "default",
      "web-app"
    ],
    "metadata": {
      "role_name": "hcvop"
    },
    "lease_duration": 2764800,
    "renewable": true,
    "entity_id": "58a97ed8-0003-c32f-b061-3ef9817a628e",
    "token_type": "service",
    "orphan": true,
    "mfa_requirement": null,
    "num_uses": 0
  }
}
```


Get the resulting token by parsing the JSON response:

`jq -r '.auth.client_token'`

Userpass Auth Method

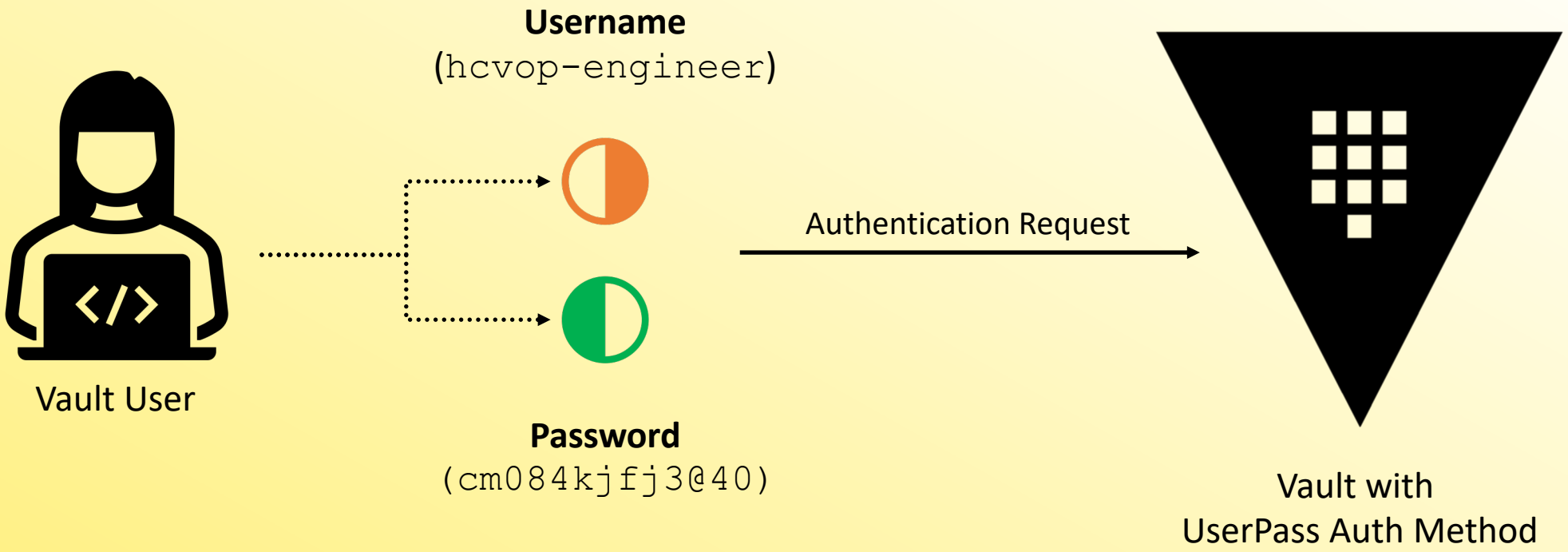
Introduction to Userpass

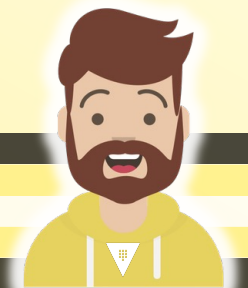
- Userpass auth method enables Vault clients to authenticate using a local username and password

- Userpass does NOT integrate or read credentials from an external identity provider. Everything is local to Vault itself.

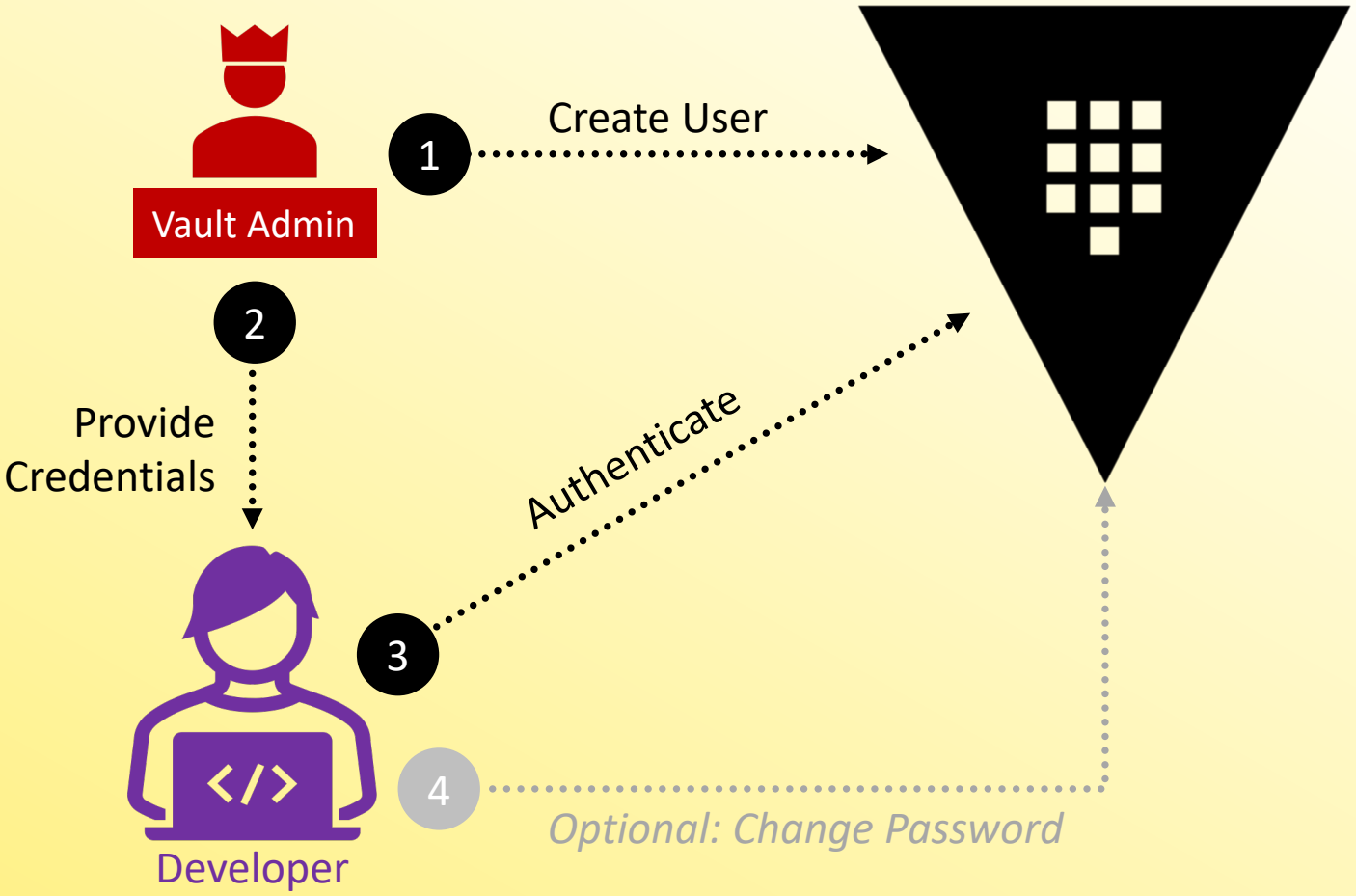
Userpass – Auth Workflow

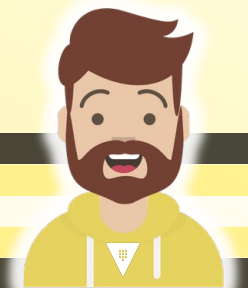
Needed for authentication





Userpass – Configuration Workflow





Enable Userpass Auth Method

```
# Enable Userpass at the default path
$ vault auth enable userpass
Success! Enabled userpass auth method at: userpass/



# Enable Userpass at a custom path
$ vault auth enable -path=vault-local userpass
Success! Enabled userpass auth method at: vault-local/
```

Create a New User

```
# Create the new user hcvop-engieer and assign a policy

$ vault write auth/userpass/users/hcvop-engineer \
    password=cm084kjfj3@40 \
   policies=engineering-policy \
   token_ttl=15m \
   token_max_ttl=8h \
Success! Data written to: auth/userpass/users/hcvop-engineer
```

Create a New User

Breaking Down the CLI Command


```
vault write auth/userpass/users/hcvop-engineer
```

- `write` — Subcomm and used to read, write, delete, or list a role
- `auth` — Default path for auth methods (not configurable)
- `/userpass` — Path where the Userpass auth method was enabled
- `/users` — Path where users are created and stored (not configurable)
- `/hcvop-engineer` — The name of the user you want to create, modify, or delete

Create a New User

A Few Configuration Tips for Userpass

- You can set the TTL of the resulting token:
  - `token_max_ttl=24h`
  - `token_ttl=1h`
- You can change the token type to be a batch token:
  - `token_type=batch`
- You can configure the token to be a use-limited token:
  - `token_num_uses=5`
- You can configure CIDR restrictions for a token:
  - `token_bound_cidrs="10.1.16.0/16"`

These are configuration parameters for a Userpass user

Read a User Configuration

```
# Read the current configuration of the hcvop-engineer user

$ vault read auth/userpass/users/hcvop-engineer
Key                        Value
---                        -----
policies                   [engineering-policy]
token_bound_cidrs          []
token_explicit_max_ttl     0s
token_max_ttl              8h
token_no_default_policy    false
token_num_uses             0
token_period               0s
token_policies             [engineering-policy]
token_ttl                  15m
token_type                 default
```

Authenticate with Userpass Credentials

```
# Authenticate with hcvop-engineer user

$ vault login -method=userpass username=hcvop-engineer
Password (will be hidden):

Success! You are now authenticated. The token information displayed below
is already stored in the token helper. You do NOT need to run "vault login"
again. Future Vault requests will automatically use this token.

Key                    Value
---                    -----
token                  hvs.CAESIKcXGsfBA8pzKwBgAQ1XferekT1n9S5PC0mgz36EKHy5zR2RFR
token_accessor         2Pm2ybDoAyRqrrhMEfDi674N
token_duration         15m
token_renewable        true
token_policies         ["default" "engineering-policy"]
identity_policies      []
policies               ["default" "engineering-policy"]
token_meta_username    hcvop-engineer
```

Update a Password for a User

```
# Authenticate with hcvop-engineer user

$ vault write auth/userpass/users/hcvop-engineer/password password=xmeij9dk20je
Success! Data written to: auth/userpass/users/hcvop-engineer/password
```

Token Auth Method

Introduction to Tokens

- Tokens are the core method for authentication
  - Most operations in Vault require an existing token (not all, though)
  - Accessing a login path doesn't, for example
- The token auth method is responsible for creating and storing tokens
  - The token auth method cannot be disabled
  - Tokens can be used directly, or they can be used with another auth method
  - Authenticating with an external identity (e.g. LDAP) dynamically generate tokens
- Tokens have one or more policies attached to control what the token is allowed to perform

Introduction to Tokens

- `service` tokens are the <u>default</u> token type in Vault
  - They are persisted to storage (heavy storage reads/writes)
  - Can be renewed, revoked, and create child tokens
  - <u>Most often, you'll be working with service tokens</u>

- `batch` tokens are encrypted binary large objects (blobs)
  - Designed to be lightweight & scalable
  - They are <u>NOT</u> persisted to storage, but they are not fully-featured
  - Ideal for high-volume operations, such as encryption
  - Can be used for DR Replication cluster promotion as well

Create a Periodic Token

```
$ vault token create -policy=hcvop -period=24h

Key                  Value
---                  -----
token                hvs.CAESINq3yTGLYZofP7iZBStz3zAktvOHfWBigN
token_accessor       fy9Jjse9SRTLIYLUfysE6qP0
token_duration       24h
token_renewable      true
token_policies       ["default" "hcvop"]
identity_policies    []
policies             ["default" "hcvop"]
```

No Max TTL – You can renew a periodic token an infinite number of times

Create a Use-Limited Token

```
$ vault token create -policy="hcvop" -use-limit=2
Key                  Value
---                  -----
token                hvs.CAESIFu4Z5VF8AWeTgAXO89WCewUqZGVUdl8Pj
token_accessor       by7TufOOtCoUZRL86PRu0cil
token_duration       768h
token_renewable      true
token_policies       ["default" "hcvop"]
identity_policies    []
policies             ["default" "hcvop"]
```

Can only use this token **TWO** times and then it will be automatically revoked

Create a Orphan Token

```
$ vault token create -policy="hcvop" -orphan
Key                  Value
---                  -----
token                hvs.CAESIMQJcAGEXN93hLZfgPWJtsMbClKhdiF8jFP
token_accessor       wAtiKTSt1PmnL2zhL537FHBa
token_duration       768h
token_renewable      true
token_policies       ["default" "hcvop"]
identity_policies    []
policies             ["default" "hcvop"]
```

Token is NOT affected by the TTL/revocation of its parent token

Set the Token Type at the Auth Method

To configure the AppRole auth method to generate batch tokens:

TERMINAL

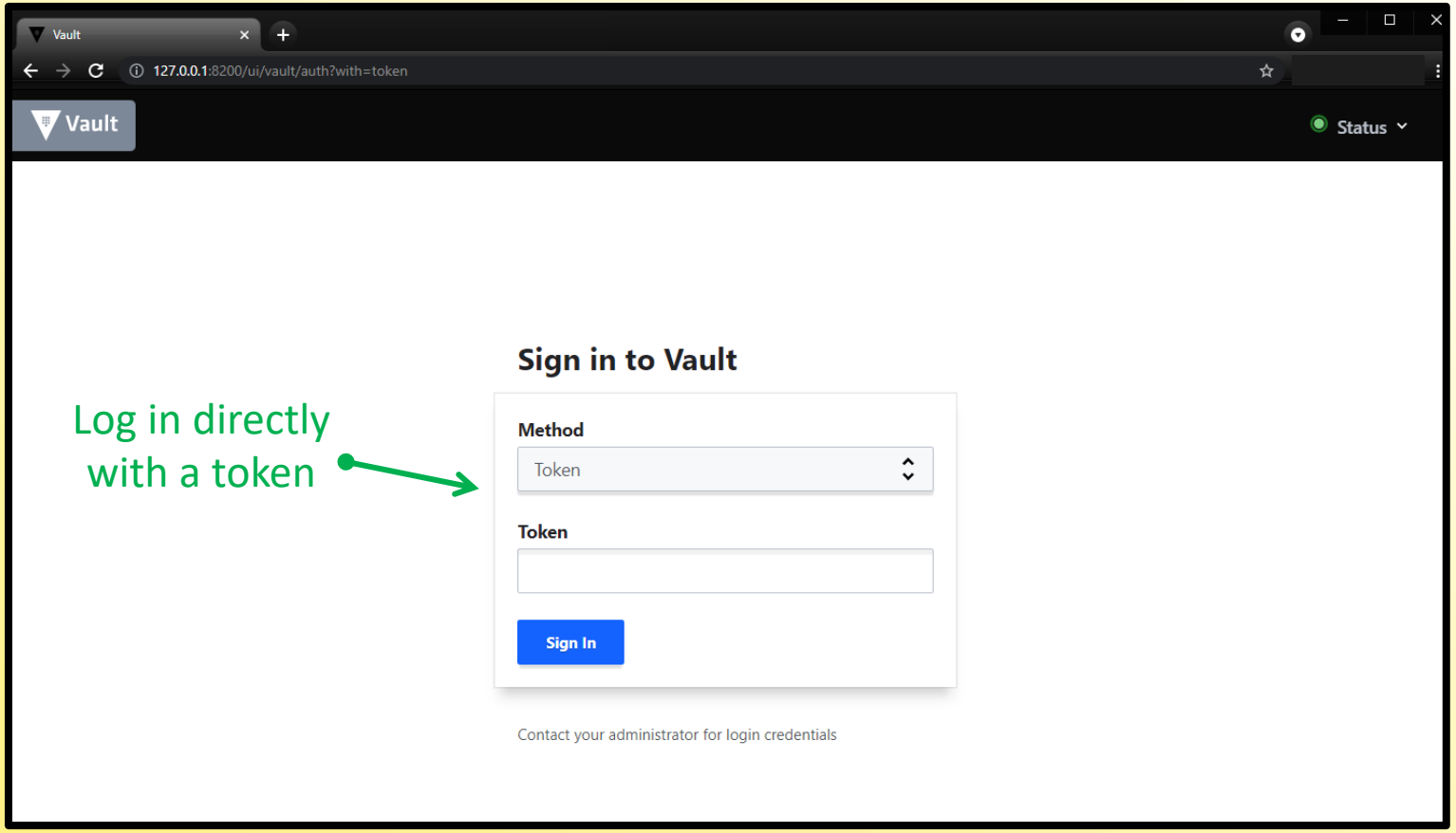
```
$ vault auth enable approle

$ vault write auth/approle/role/hcvop policies="engineering" \
        token_type="batch" \
        token_ttl="60s"
```

To configure the AppRole auth method to generate periodic tokens:

TERMINAL

```
$ vault write auth/approle/role/hcvop policies="hcvop" \
        period="72h"
```

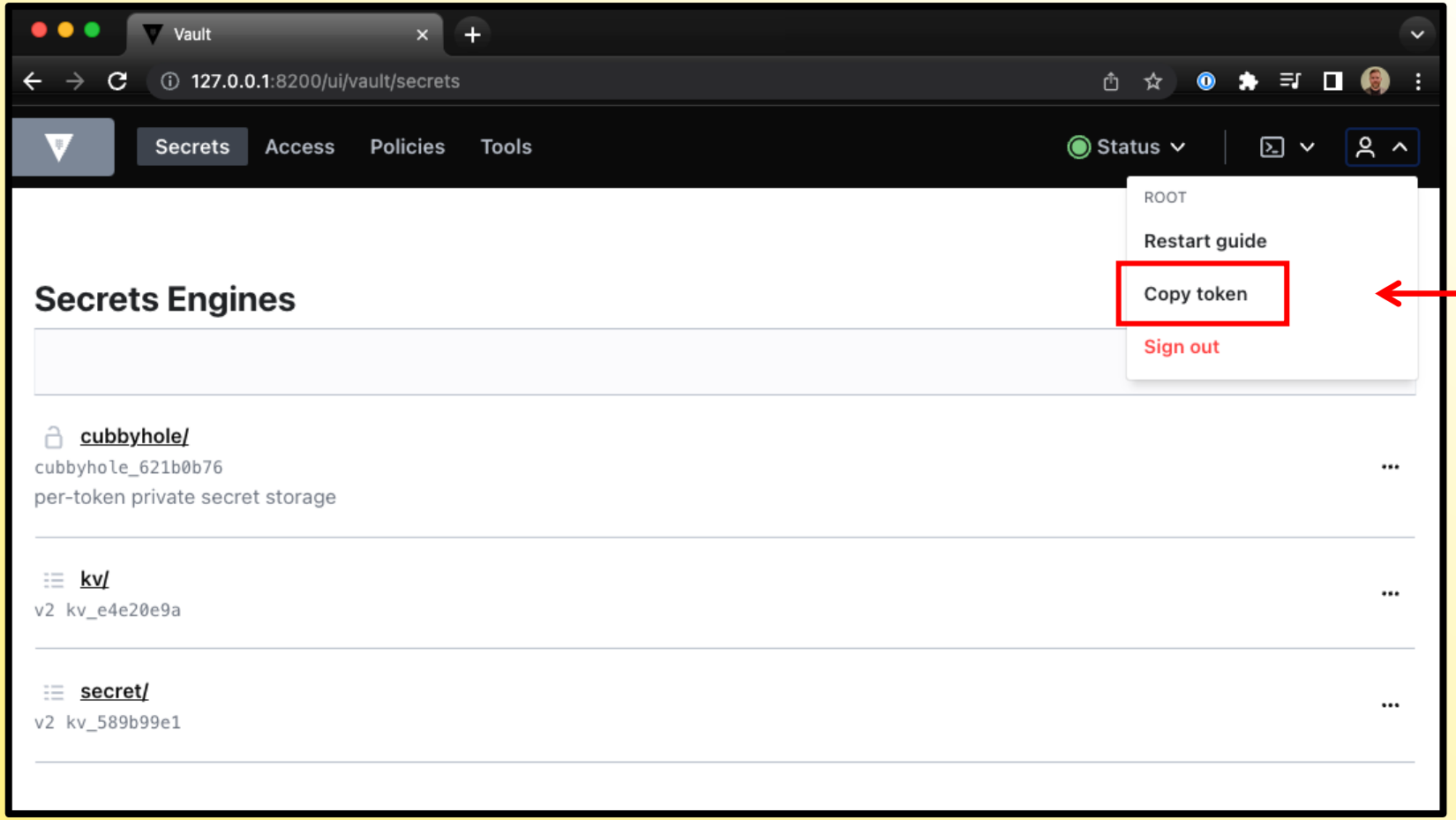
Authenticate with a Token



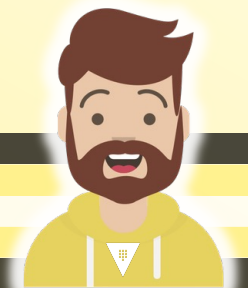
Log in directly with a token

Authenticate with a Token

Vault
CERTIFIED
OPERATIONS
PROFESSIONAL



Copy the Token
You are Using



Authenticate with a Token

Client token must be sent in the X-Vault-Token HTTP header

- Note that `Authorization: Bearer <token>` is valid as well

Terminal

```
$ curl --header "X-Vault-Token: hvs.cDIPyitdJKSm46ydTXJOsaQR" \
       --request POST \
       --data '{ "apikey": "3230sc$832d" }'
       https://vault.hcvop.com:8200/v1/secret/apikey/splunk
```

Terminal

```
$ curl --header "Authorization: Bearer hvs.cDIPyitdJKSm46ydTXJOsaQR" \
       --request GET \
       https://vault.hcvop.com:8200/v1/secret/data/apikey/splunk
```

Authenticate with a Token

```
$ vault login
Token (will be hidden):
Success! You are now authenticated. The token information displayed below
is already stored in the token helper. You do NOT need to run "vault login"
again. Future Vault requests will automatically use this token.

Key                  Value
---                  -----
token                hvs.cDIPyitdJKSm46ydTXJOsaQR
token_accessor       ggVElRJkK0mWS2uYt9Kdi0SL
token_duration       24h
token_renewable      false
token_policies       ["engineering"]
identity_policies    []
policies             ["engineering"]
```

Revoke a Token

Tokens can easily be revoked so they can no longer be used for authentication

- **This includes a root token**

```
$ vault token revoke hvs.cDIPyitdJKSm46ydTXJOsaQR
Success! Revoked token (if it existed)
```