Compare Authentication Methods

Section Overview

Objective 1a

Describe Auth Methods

- Introduction to Auth Methods and Terms
- Authentication Workflow
- Entities and Groups

Objective 1b

Choose an Auth Method based on Use Case

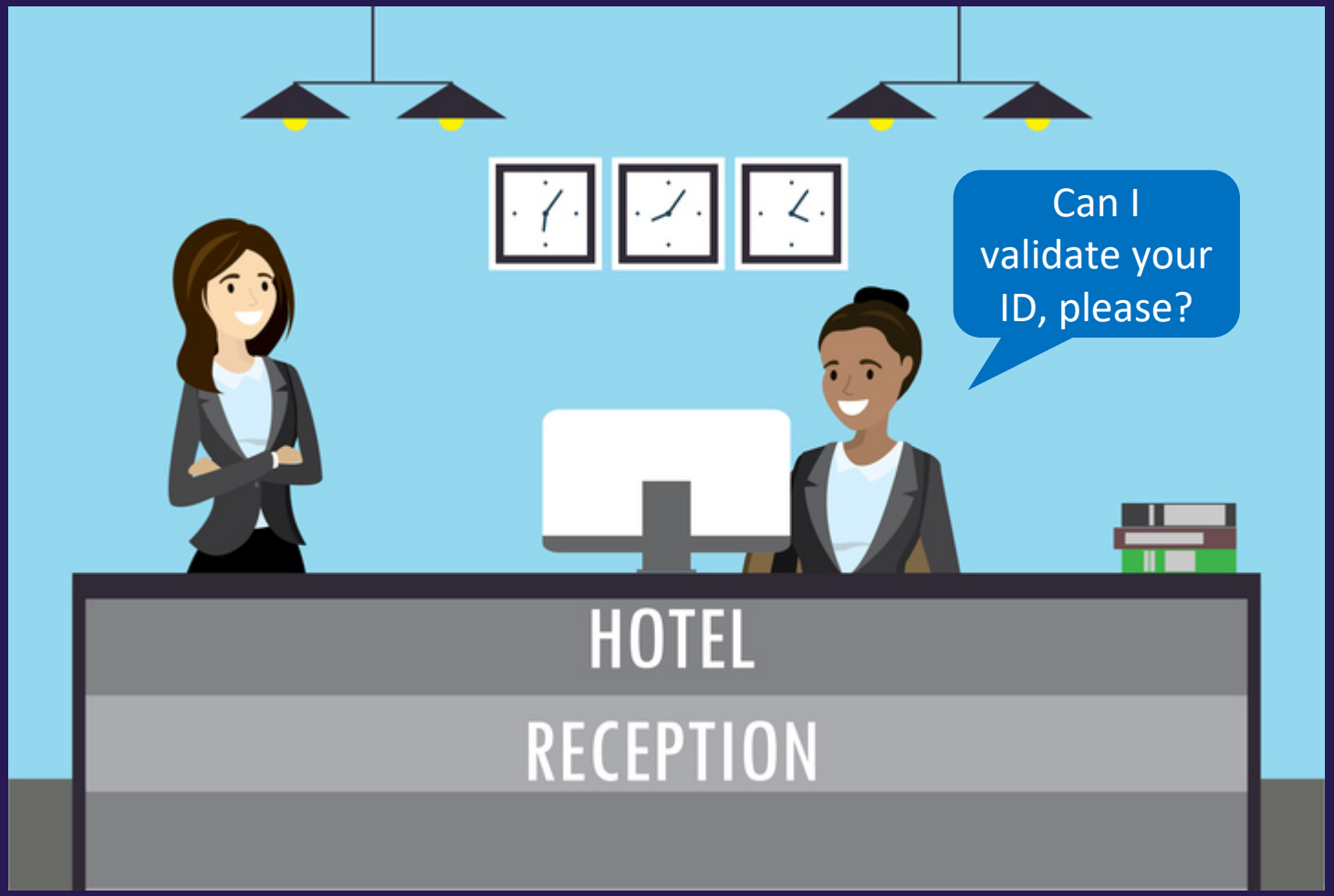
- Automation Use Cases
- Cloud-Based Requests
- UI/CLI Authentication

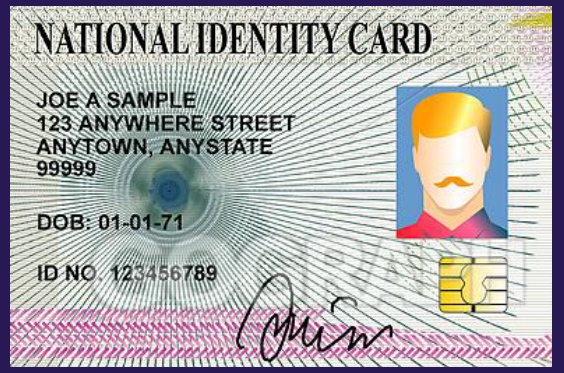
Objective 1c

Differentiate human vs. system auth methods

- Human-Based Access
- Cloud-Based Access
- Machine to Machine Access

Vault Authentication



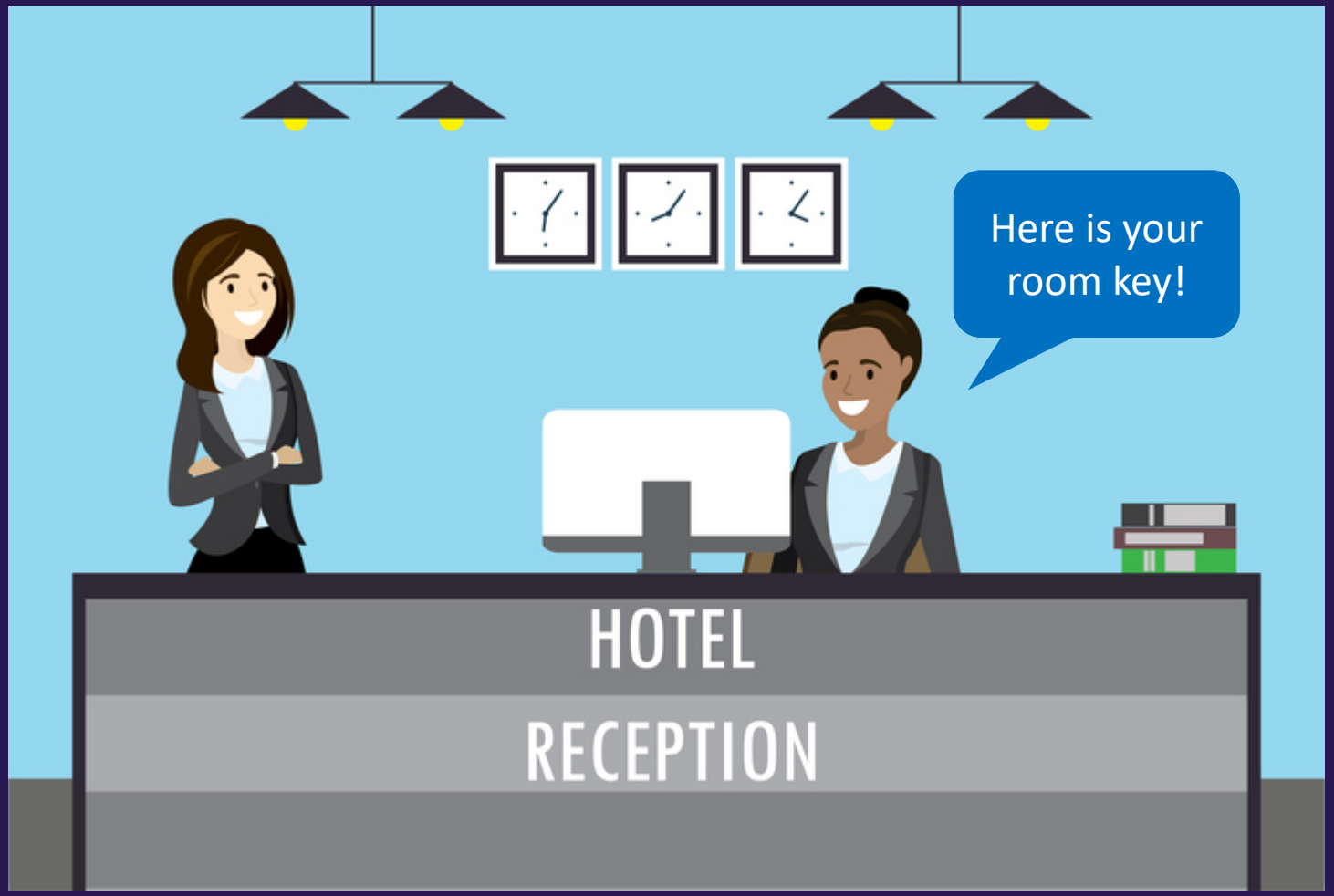


Vault Authentication

Authentication

Vault Authentication

VALID FOR 3 DAYS



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

"The fundamental goal of an auth method is to obtain a Vault token.

Future Requests to Vault will be made using the resulting token

Auth Methods

- Tokens are the core method for authentication within Vault
  - Most operations in Vault require an existing token

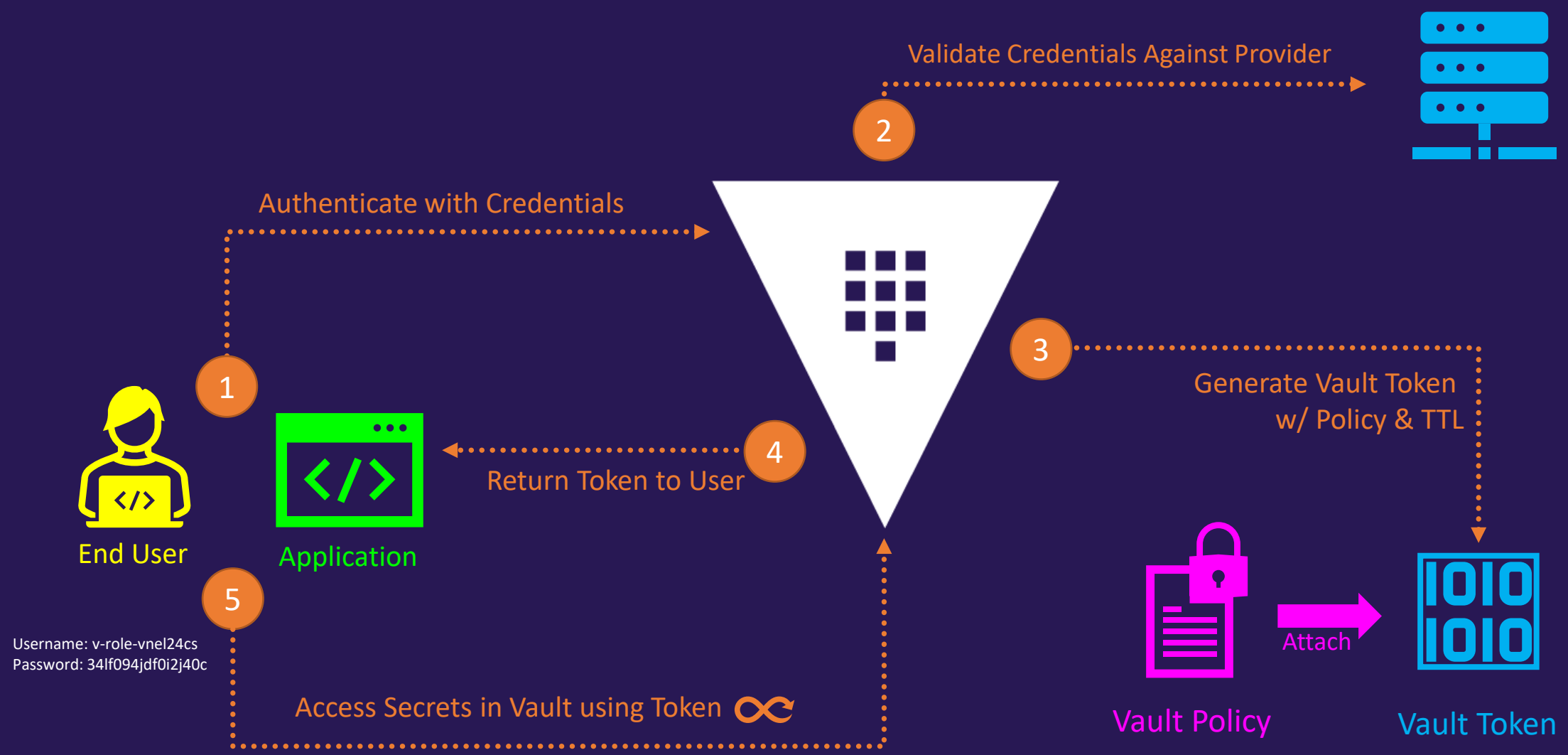
- The token auth method is responsible for creating and storing tokens
  - Token auth method cannot be disabled
  - Authenticating with external identity (LDAP, OIDC) will generate a token

More on Vault tokens in Objective 3

Auth Methods

- If you do not supply a token for non-authentication requests, there are no redirects for authentication – no hints or suggestions – you will get a 403 Access Denied Error.

Auth Methods - Workflow



Auth Methods

Auth Methods

- Most auth methods must be enabled 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
  - Only method of authentication for a new Vault deployment is a root token

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.

Auth Methods

Each auth method is enabled at a path

- You can choose the path name when (and only when) you enable the auth method
- If you do not provide a name, the auth method will be enabled at its default path
  - The default path name is equal to the type of auth method (i.e., aws is mounted at "aws", approle is mounted at "approle")

Terminal

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

Configuring Auth Methods

Command Line Interface (CLI)

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
----              ----       --------                 -----------
bryan/            approle    auth_approle_d8c20abe    n/a
token/            token      auth_token_89ce3371      token based credentials
vault-course/     approle    auth_approle_b3f0c92d    n/a
```

Configuring Auth Methods

Enable Auth Method

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=vault-course approle
Success! Enabled approle auth method at: vault-course/
```

More specifics on working with the Vault CLI in Objective 6

Configuring Auth Methods

Command Line Interface (CLI)

```
$ vault auth enable approle
```

Type of Vault object you want to work with

Subcommand

Type of Auth Method

Configuring Auth Methods

Command Line Interface (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

Configuring Auth Methods

Command Line Interface (CLI)

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

Type of Vault object you want to work with

Sub-command

Customize the Path Name

Add a description

Type of Auth Method

Configuring Auth Methods

Command Line Interface (CLI)

```
$ vault auth disable apps
```

Type of Vault object you want to work with

Sub-command

Path of the Object to be disabled

Auth Methods

Command Line Interface (CLI)

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/vault-course \
    secret_id_ttl=10m \
    token_num_uses=10 \
    token_ttl=20m \
    token_max_ttl=30m \
    secret_id_num_uses=40
```

Brief Intro to the Vault API

Vault offers a fully-featured API intended for machine-to-machine interaction

Critical components of an API request that need to be included:

- ✓ The request type (GET, POST, DELETE)
- ✓ The appropriate headers (X-Vault-Token, Authorization, X-Vault-Namespace)
- ✓ The data (if required)
- ✓ The API endpoint (what Vault component you're working with)

Brief Intro to the Vault API

HTTP API – Where Do I Need a Token?

Using an Auth Method

When you are authenticating to Vault via API, you do not need to specify a token (because you haven't retrieved one yet)

vs

Configure Auth Method

When you are enabling, configuring, or disabling an auth method, you do need to provide a token with the appropriate permissions

More specifics on working with the Vault API in Objective 8

Configuring Auth Methods

HTTP 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.example.com:8200/v1/sys/auth/approle
```

API Endpoint

Vault Authentication

Command Line Interface (CLI)

There are a few ways to authenticate to Vault when using the CLI

1. Use the `vault login` command
   - Authenticate using a token or another auth method
   - Makes use of a token helper

2. Use the `VAULT_TOKEN` Environment Variable
   - Used if you already have a token

Vault Authentication

Command Line Interface (CLI)

Using the `vault login` command

By default, vault login uses a "token" method

Terminal

```
$ vault login s.fhNBot4hRBfDWJ2jBdTwimaG
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                s.fhNBot4hRBfDWJ2jBdTwimaG
token_accessor       502YCRmp1SfZ8YCdfbYeS9fj
token_duration       ∞
token_renewable      false
token_policies       ["root"]
identity_policies    []
policies             ["root"]
```

Vault Authentication

Command Line Interface (CLI)

Using the `vault login` command

Used to obtain a token

Terminal

```
vault login -method=userpass username=bryan
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                  s.jgSgKqDOnaOxu30ffC0rZWB0
token_accessor         SpiJi6bghz4huS8MG4HsLmNp
token_duration         768h
token_renewable        true
token_policies         ["admin" "default"]
identity_policies      []
policies               ["admin" "default"]
token_meta_username    bryan
```

Got a Token!

Vault Authentication

Command Line Interface (CLI)

```
vault login -method=userpass username=bryan
```

Type of Auth Method Used to Authenticate
(not the enabled path)

Argument for Passing Additional Data

vaultproject.io/docs/commands/login#usage

Vault Authentication

Command Line Interface (CLI)

Token Helper

Caches the token after authentication. Stores the token in a local file so it can be referenced for subsequent requests

Terminal

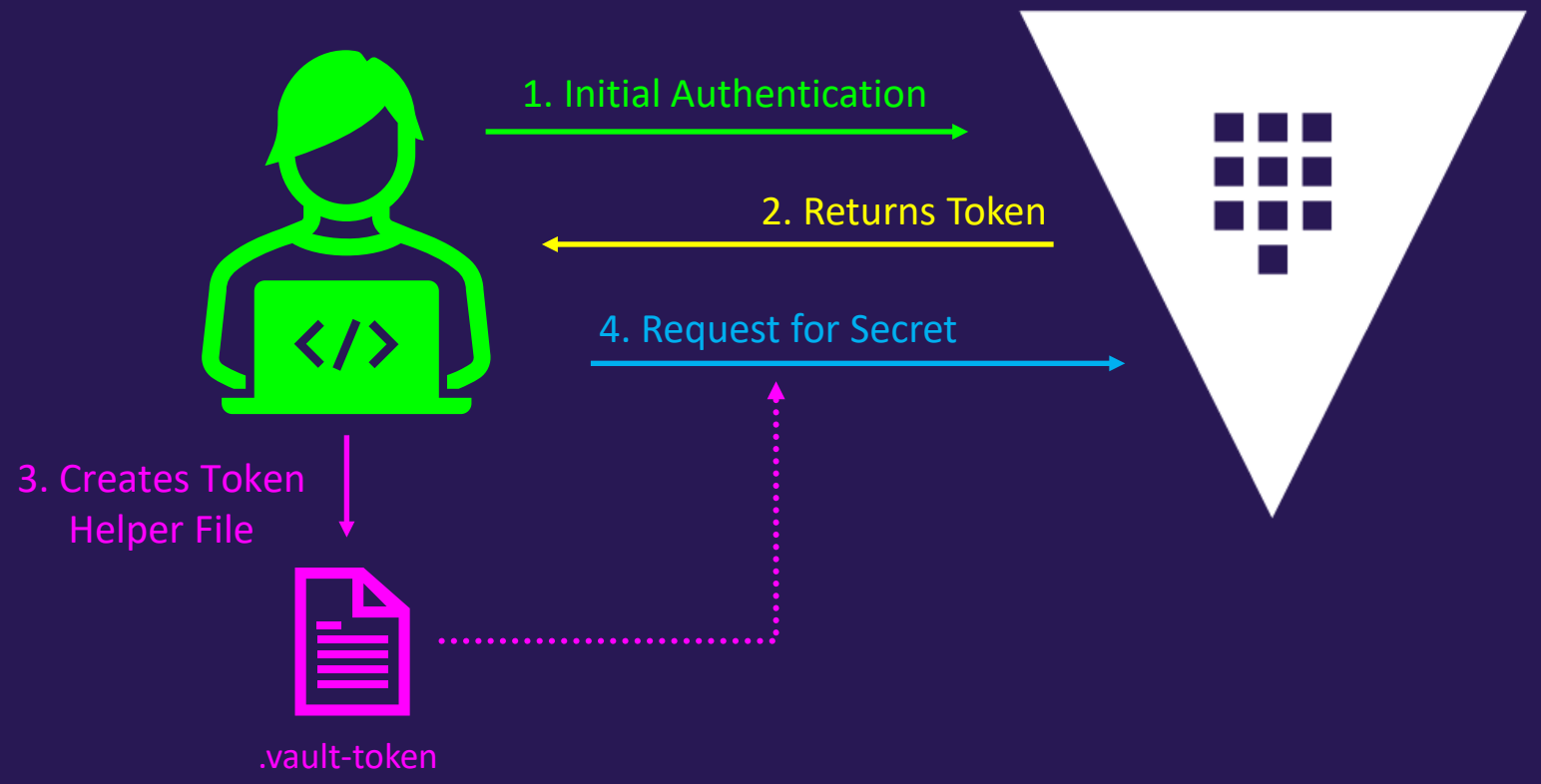
```
$ vault login s.fhNBot4hRBfDWJ2jBdTwimaG

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                s.fhNBot4hRBfDWJ2jBdTwimaG
token_accessor       502YCRmp1SfZ8YCdfbYeS9fj
token_duration       ∞
token_renewable      false
token_policies       ["root"]
identity_policies    []
policies             ["root"]
```

.vault-token

Vault Authentication

Command Line Interface (CLI)

Token Helper - Workflow



Vault Authentication

HTTP API Response

Parsing the JSON Response to Obtain the Vault Token

Terminal

```
$ export VAULT_ADDR="https://vault.example.com:8200"
$ export VAULT_FORMAT=json
$ OUTPUT=$(vault write auth/approle/login role_id="12345657" secret_id="1nv84nd3821s")
$ VAULT_TOKEN=$(echo $OUTPUT | jq '.auth.client_token' -j)
$ vault login $VAULT_TOKEN
```

Vault Authentication

HTTP API

- Authentication requests to the Vault HTTP API return a JSON response that include:
  - the token
  - the token accessor
  - information about attached policies

- It is up to the user to parse the response for the token and use that token for any subsequent requests to Vault.

Vault Authentication

HTTP API

Authenticating with an Auth Method:

- Method: POST
- Response: JSON

Contains role_id and secret_id

Terminal

```
$ curl \
    --request POST \
    --data @auth.json \
    https://vault.example.com:8200/v1/auth/approle/login
```

API Endpoint

Vault Authentication

HTTP API Response (snippet)

Terminal

Snippet of Response

```
{
  "request_id": "0f874bea-16a6-c3da-8f20-1f2ef9cb5d22",
  "lease_id": "",
  "renewable": false,
  "lease_duration": 0,
  "data": null,
  "wrap_info": null,
  "warnings": null,
  "auth": {
    "client_token": "s.wjkffdrqM9QYTOYrUnUxXyX6",
    "accessor": "Hbhmd3OfVTXnukBv7WxMrWld",
    "policies": [
      "admin",
      "default"
    ],
```

Service Token

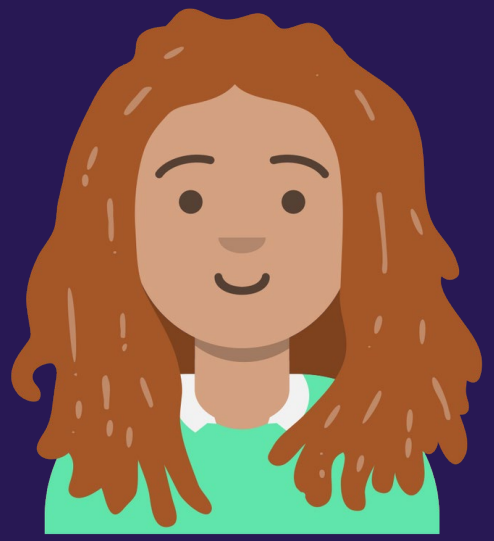
Policies associated with our token

Vault Entities

- Vault creates an entity and attaches an alias to it if a corresponding entity doesn't already exist.
  - This is done using the Identity secrets engine, which manages internal identities that are recognized by Vault
- An entity is a representation of a single person or system used to log into Vault. Each has a unique value. Each entity is made up of zero or more aliases
- Alias is a combination of the auth method plus some identification. It is a mapping between an entity and auth method(s)

Vault Entities

Julie Smith
Finance Specialist



alias

entity_id

UserPass: jsmith

Entity_ID: b81de864-75fa-5619-1fca-ddd72bbe5b29

department: finance
office: San Francisco
team: accounts-payable

accounting

metadata

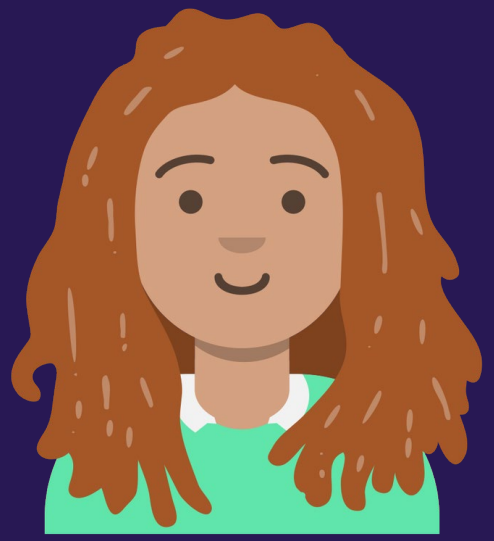
policy

UserPass

Vault Entities

Julie Smith

Finance Specialist



UserPass: jsmith
Entity_ID: b81de864-75fa-5619-1fca-ddd72bbe5b29

department: accounting
sub-team: accounts-payable

accounting

LDAP: jsmith@example.com
Entity_ID: e93d24b2a-b894-0998-43ce-4294cb9ea9b

department: finance
team: management

finance

GitHub: jsmith22
Entity_ID: 4c9ed3482-4894-ced9-a1b2-90344be93aa

location: us
sales-region: west

accounts payable

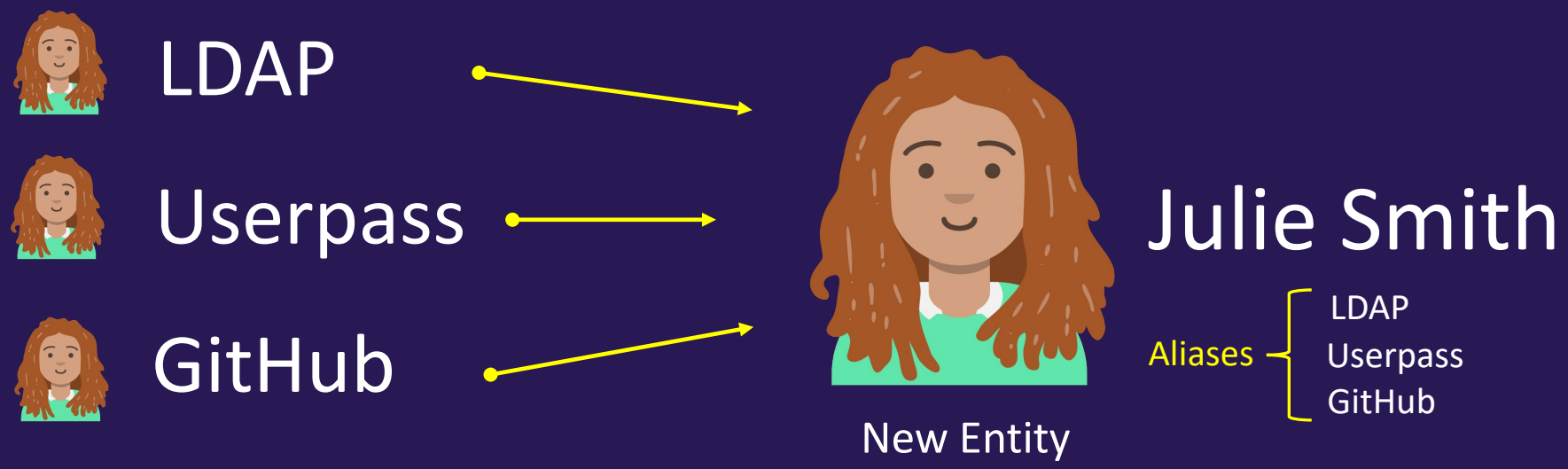
Auth Options:

UserPass
LDAP
GitHub

Vault Entities

Consolidating Logins under a Single Entity

- An entity can be manually created to map multiple entities for a single user to provide more efficient authorization management
- Any tokens that are created for the entity inherit the capabilities that are granted by alias(es).



Vault Entities

Entity

Name: Julie Smith
Entity_ID: e48de234-58fa-0093-5fde-e5b99abe8b33
Policy: *management*

Aliases:

GitHub: jsmith22
Entity_ID: 4c9ed3482-4894-ced9-a1b2-90344be93aa
Policy: *finance*

LDAP: jsmith@example.com
Entity_ID: e93d24b2a-b894-0998-43ce-4294cb9ea9b
Policy: *accounting*

UserPass: jsmith
Entity_ID: b81de864-75fa-5619-1fca-ddd72bbe5b29

Aliases

Vault Entities

Name: Julie Smith
Entity_ID: e48de234-58fa-0093-5fde-e5b99abe8b33
Policy: *management*

Aliases:

GitHub: jsmith22
Entity_ID: 4c9ed3482-4894-ced9-a1b2-90344be93aa
Policy: *finance*

LDAP: jsmith@example.com
Entity_ID: e93d24b2a-b894-0998-43ce-4294cb9ea9b
Policy: *accounting*

jsmith@example.com

2. Validate with LDAP

1. Authenticate with LDAP credentials

3. Return a Vault token

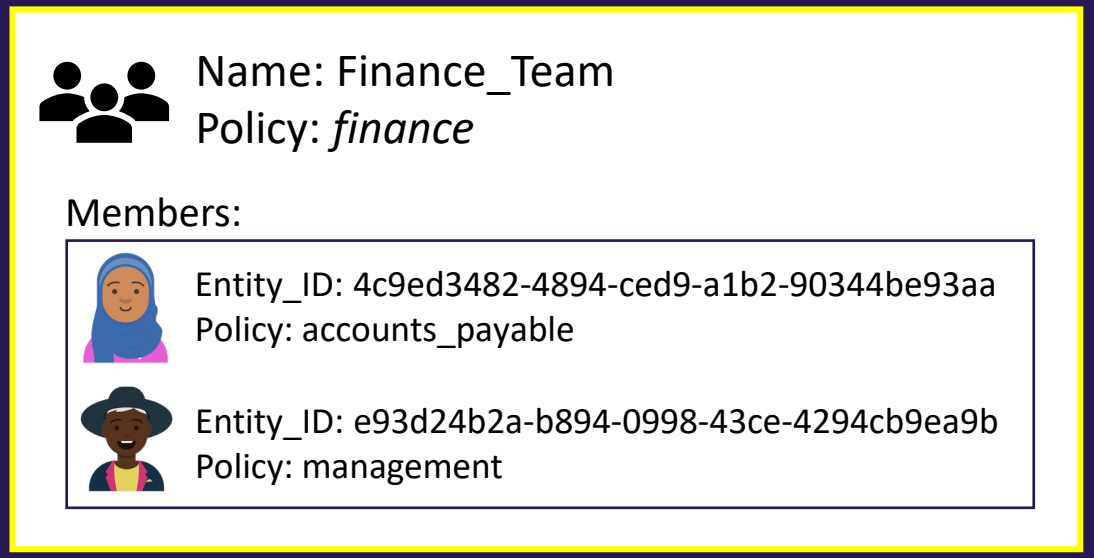
TOKEN

Policies
accounting
management

Token inherits **capabilities** granted by both policies

Vault Groups

- A group can contain multiple entities as its members.
- A group can also have subgroups.
- Policies can be set on the group and the permissions will be granted to all members of the group.



Vault Groups

Name: Finance_Team
Policy: *finance*

Members:

Name: Maria Shi
Entity_ID: 4c9ed3482-4894-ced9-a1b2-90344be93aa
Policy: *accounts_payable*

Entity Aliases:

Username: maria.shi
Policy: *base-user*

Name: John Lee
Entity_ID: e93d24b2a-b894-0998-43ce-4294cb9ea9b
Policy: *management*

Entity Aliases:

Username: john.lee
Policy: *super-user*

TOKEN

Policies
super-user
management
finance

Token inherits **capabilities** granted by alias, entity, and the group

Vault Groups

Internal Group

Groups created in Vault to group entities to propagate identical permissions

Created Manually

External Group

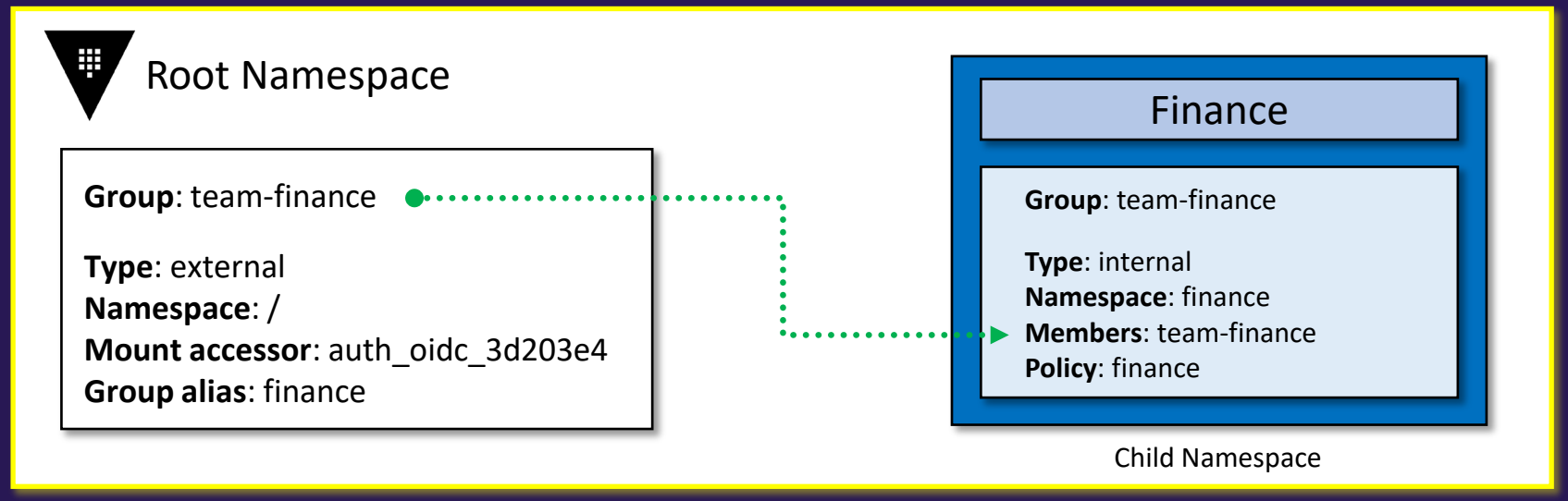
Groups which Vault infers and creates based on group associations coming from auth methods

Created Manually or Automatically

Vault Groups

Internal Groups

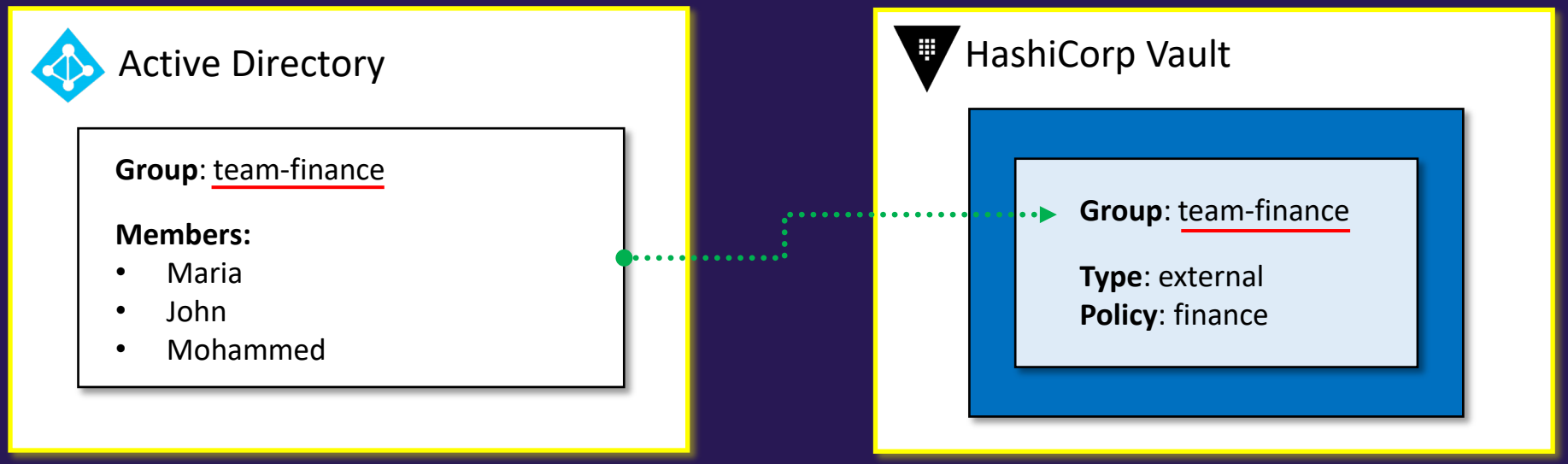
- Internal groups can be used to easily manage permissions for entities
- Frequently used when using Vault Namespaces to propagate permissions down to child namespaces
  - Helpful when you don't want to configure an identical auth method on every single namespace



Vault Groups

External Groups

- External groups are used to set permissions based on group membership from an external identity provider, such as LDAP, Okta, or OIDC provider.
- Allows you to set up once in Vault and continue manage permissions in the identity provider.
  - Note that the group name must match the group name in your identity provider



Choosing an Auth Method

- Certain auth methods are ideal for different use cases
- Many auth methods may satisfy the requirements, but often there's one that works "the best" for a situation

- In contrast, just because you are using a certain platform does not mean you need to use the related auth method
  - **Example:** Azure virtual machines can authenticate using the Azure auth method, but AppRole, Userpass, TLS, OIDC, etc. would still be a possibility
    - Azure *might* be the best but you're not limited to only Azure.

- It's usually easy to eliminate auth methods based on the way they operate or integrate with applications

Choosing an Auth Method

Key words when choosing an auth method:

- Frequently Rotated
  - generally means a dynamic credential
  - Meets the requirements: AWS, LDAP, Azure, GCP, K8s
  - Does not meet the requirements: Userpass, TLS, AppRole
- Remove Secrets from Process or Build Pipeline
  - generally means a dynamic or integrated credential
  - Meets the requirements: AWS, Azure, GCP, K8s
  - Does not meet the requirements: Userpass, LDAP

Choosing an Auth Method

- Use Existing User Credentials
  - Generally means you should integrate with an existing Identity Provider
  - Meets the Requirement: OIDC, LDAP, Okta, GitHub
  - Does Not Meet the Requirements: Userpass, AWS, Azure, GCP

Differentiate Human vs. System Auth Methods

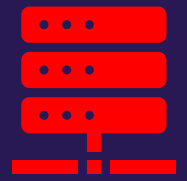
Vault offers many different types of auth methods

Many are intended for human-based authentication, while many are geared towards machine-to-machine authentication

Let's break these down….

Auth Methods

kubernetes

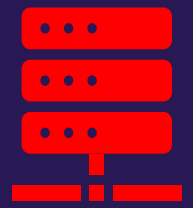


Auth Methods

Human-based Auth Methods

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 Auth Methods

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

aws

TOKENS

CLOUDFOUNDRY

TLS Certificates

Kerberos

ORACLE CLOUD

AppRole

Alibaba Cloud

Microsoft Azure

kubernetes

Exam Tips for Objective 1

Exam Tips

- Primary functionality of auth methods = validate/manage identities and issue tokens
- Tokens are tied to a policy that permits access to secrets
- The ultimate goal of an auth method is to obtain a token
- Know the auth methods that Vault supports

Exam Tips

- Understand how most of the auth methods work at a high level to help determine the correct auth method for different use cases
- Remember the key words (frequently rotated, use existing identity provider, etc.)
- Remember that you are not restricted to only the auth method where the workload is running

Exam Tips

- Know the human-based auth methods (LDAP, OIDC, GitHub, etc.)
  - Anything that is interactive requiring hands-on keyboard
- Know the system-based auth methods (AWS, Azure, TLS, AppRole, etc.)
  - Auth methods using complex credentials or platform-based creds)

END OF SECTION