Vault

Implementation Foundations

Module 10: Policies

What You Will Learn

Policies

- Policy Overview
- Tokens and Polices
- Writing Polices
- Associating Policies

Policies

Policies

- Vault Policies control what entities can do on any given API endpoint
- A policy grants CRUD capabilities
- Is the Role Based Access Control system for Vault
- Are written in HCL

```
path "auth/token/lookup-self" {
    capabilities = ["read"]
}

path "auth/token/renew-self" {
    capabilities = ["update"]
}
```

Policies and Tokens

```
$ vault write auth/ldap/groups/sre policies="dev, ops"

$ vault token create -policy="admin"
```

A list of associated polices can be specified when

- Adding a new Auth Method
- Configuring a specific user
- Adding a new Secret Engine

Policies

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

- Everything within Vault is path based accessible
- These paths are how we logically access secrets, configuration and more
- The same paths are how we control access to those secrets, configuration and more

Policies - Paths

Here are some common examples

- `/<root_path>/config` - configures backend level settings for this particular path
- `/<root_path>/data/example` - creates a new version of a secret at the specified location.
- `/<root_path>/roles/example` - creates or updates a role definition
- `/<dynamic_path>/creds/example` - this endpoint generates a new set of dynamic credentials based on the named role

Policies - Permission

- **create (POST/PUT)** - Allows creating data at the given path
- **read (GET)** - Allows reading the data at the given path
- **update (POST/PUT)** - Allows changing the data at the given path
- **delete (DELETE)** - Allows deleting the data at the given path
- **list (LIST)** - Allows listing values at the given path
- **sudo** - Allows access to paths that are root-protected
- **deny** - Explicit deny, Overrides any other permission

Default Policies

There are two starting policies when you initialize and unseal Vault:

- **root policy** - superuser with all permissions
- **default policy** - least privileged functional permissions

When a new entity is created (user, system, etc) they are automatically given the default policy

Writing Policies - Best Practices

When trying to build fine-grained and least-privileged policies the first step is to gather requirements

Gather Policy Requirements:

- What is the entity (Person, Team, Application, System)
- What do they need inside Vault

Practice least privileged and fine-grained policies

- Test to make sure that the policies do not grant too much or too little permissions
- Iterate, Edit and Repeat!!!
- Version control and audit your policies
    - Using a VSC backend and pipeline to manage your policies

Gathering Requirements

- What is the Entity you are giving access to?

- How would they/it authenticate?

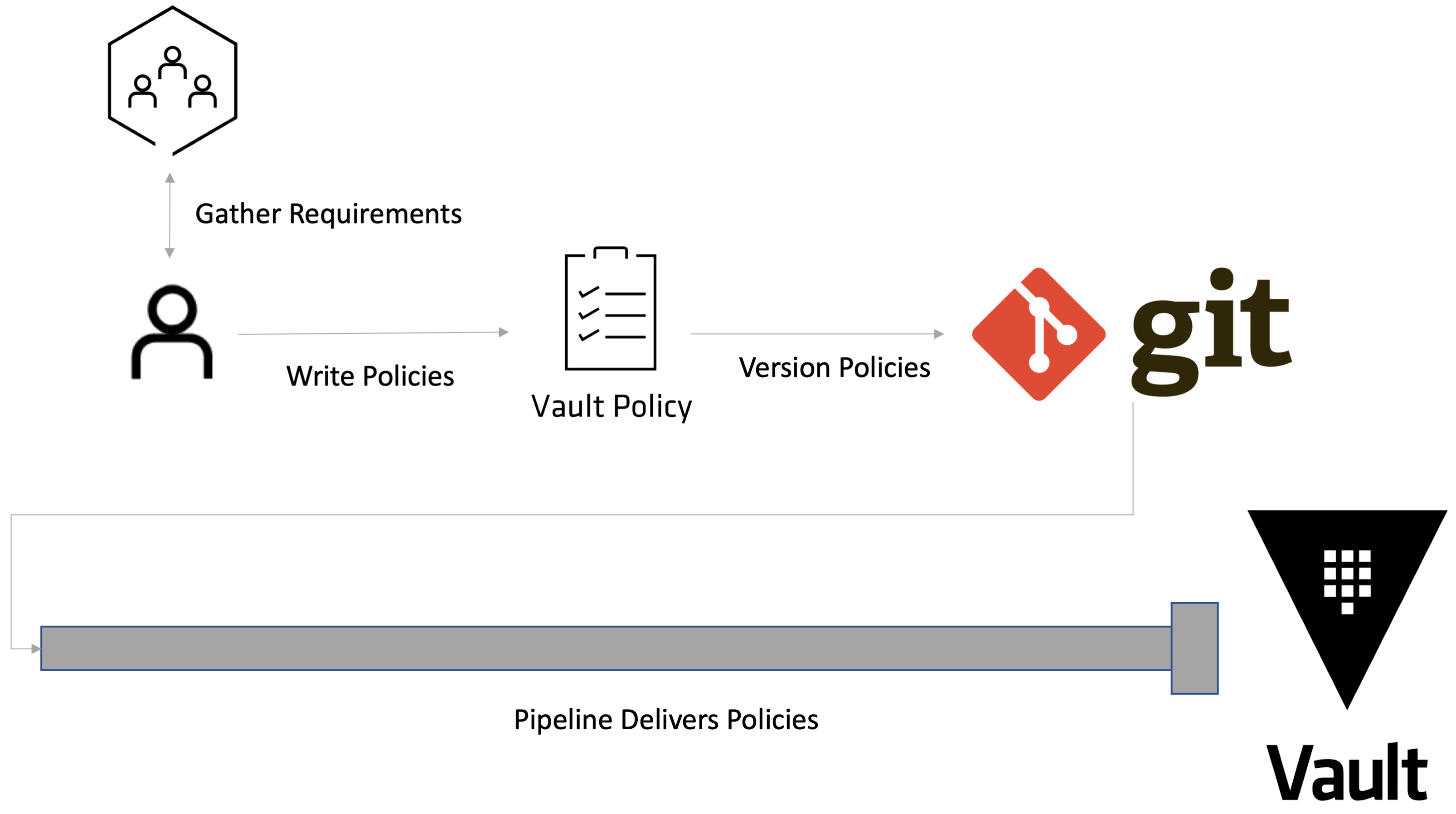
    - Person - AD/LDAP, Github, Okta, Username and Password
    - System - AWS, Azure, GCP, K8s, PCF, TLS
    - Application - AppRole, TLS, Token

Gathering Requirements

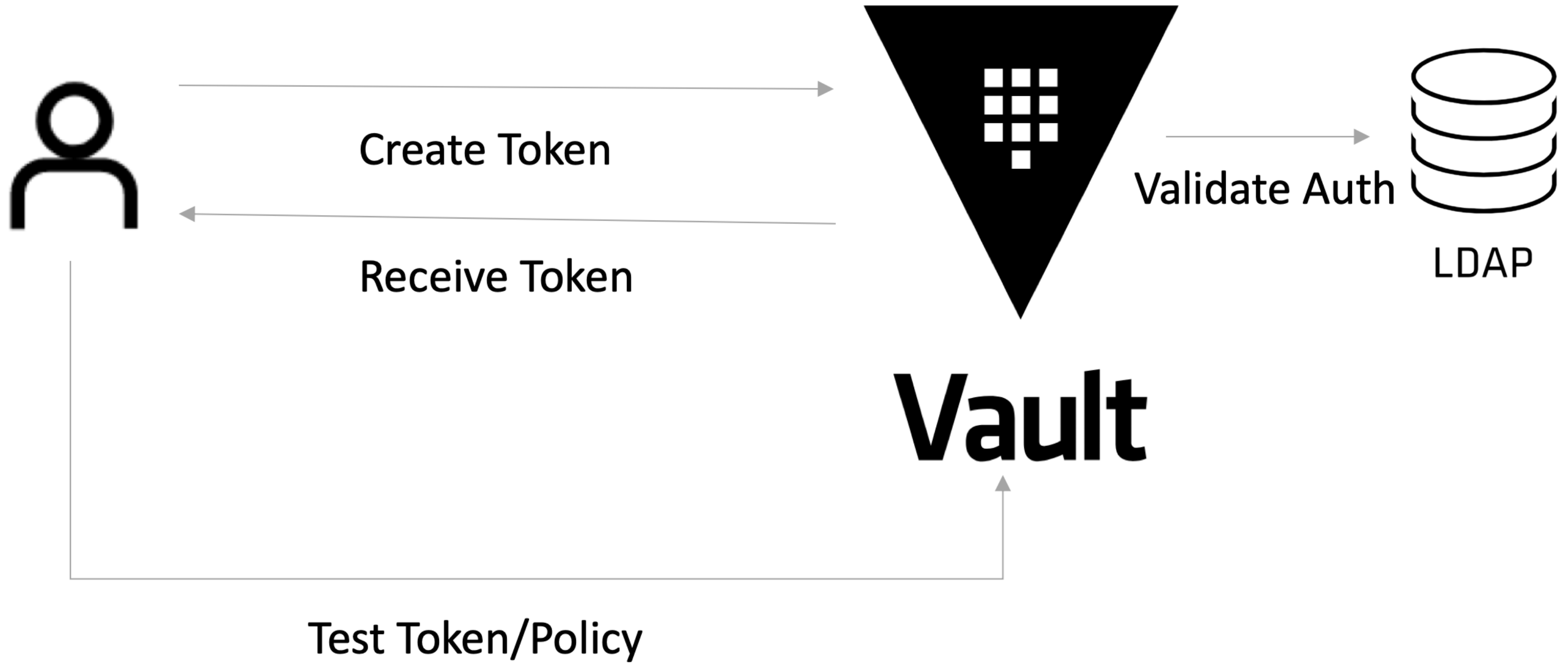
- What does that entity need access to?
- What are the minimal permissions it needs?

```
$ path "sys/*" {capabilities = ["create", "read", "update", "delete", "list", "sudo"]}
$ path "vault read database/creds/my-role" {capabilities = ["create", "read", "update"]}
$ path "/sys/audit" {capabilities = [read", "sudo"]}
```

Developing Policies



Policy Life Cycle



Policies Association

- Vault can automatically associate a set of policies to a token based on an authorization
- This configuration varies significantly between authentication backends
- This creates an authentication mapping to the policy
- When an entity authenticates successfully to Vault they will be given a token which has the list of policies attached

Policies Association (cont.)

```
vault write \
   auth/userpass/users/bwallace \
   password="s3cr3t!" \
   policies="dev-readonly,logs"
```

After successful authentication:

```
Key                     Value
--                       --
token                   mellon
token_accessor          fr!3nD20
token_renewable         false
token_policies          ["readonly,logs"]
policies                ["readonly,logs"]
```

Chapter Summary

- Policies are code and should be treated as such
    - Iterated on
    - Versioned
    - Tested
- Policies map to authentication
- Policies map to paths within Vault
- Don't edit policies in the GUI

Reference links

- Continue to Learn Policies
- More Policy Knowledge
- Guides
- Concept Review of Tokens

Vault Policies Module Complete!