Secure Multi-Tenancy with Namespaces

Graham Land

EMEA, Customer Success Manager at HashiCorp

HashiCorp

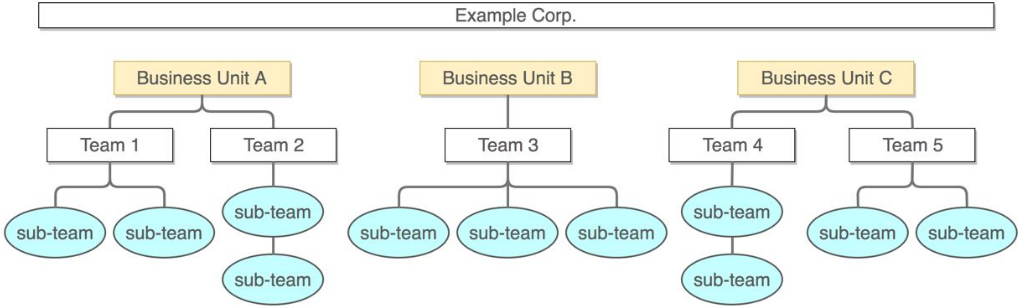
Agenda

- Introduction to Namespaces
- Writing Policies for Namespaces
- Identities across Namespaces
- EA Example
- Demo

Namespaces

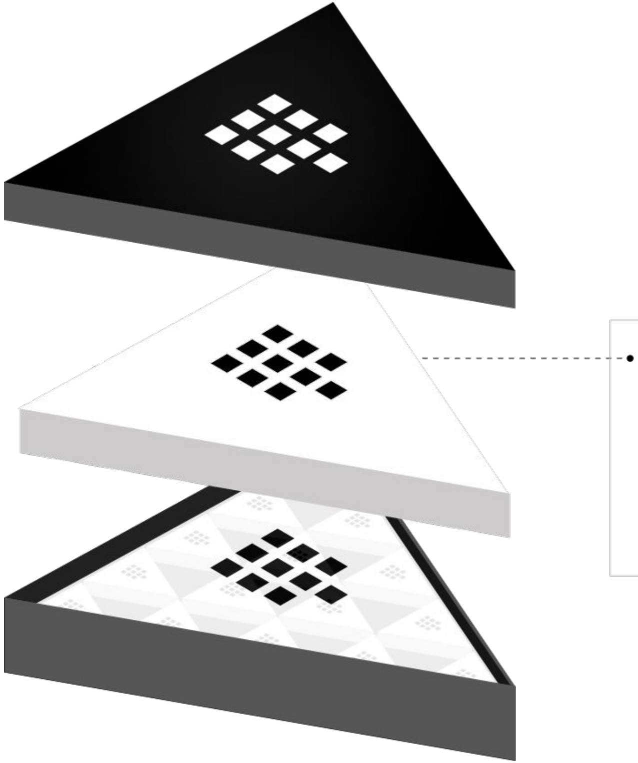
What is Namespaces?

- Namespaces is a **Vault Enterprise** feature where you can create an ***isolated*** space for each tenant (organization, team, application, etc.) to work in
- Each namespace can have its own:
  - Policies
  - Secret Engines
  - Auth Methods
  - Tokens
  - Identity entities and groups



NOTE: Identity groups can pull in entities and groups from other namespaces.

Root Namespace



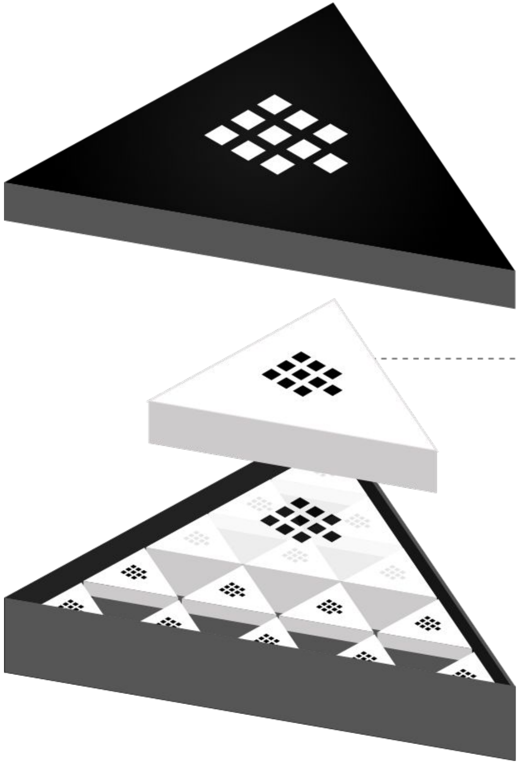
Root (Namespace)

Members:
Security Team

Namespace Specific Configuration:
Defined global member access
Defined global authentication mounts
Defined global secrets engines

Note: Vault supports namespaces within namespaces. By default there can always be a globally managed namespace that has rights to sub-namespaces, such as the Teams, and smaller namespaces

Namespaces for Teams & Groups



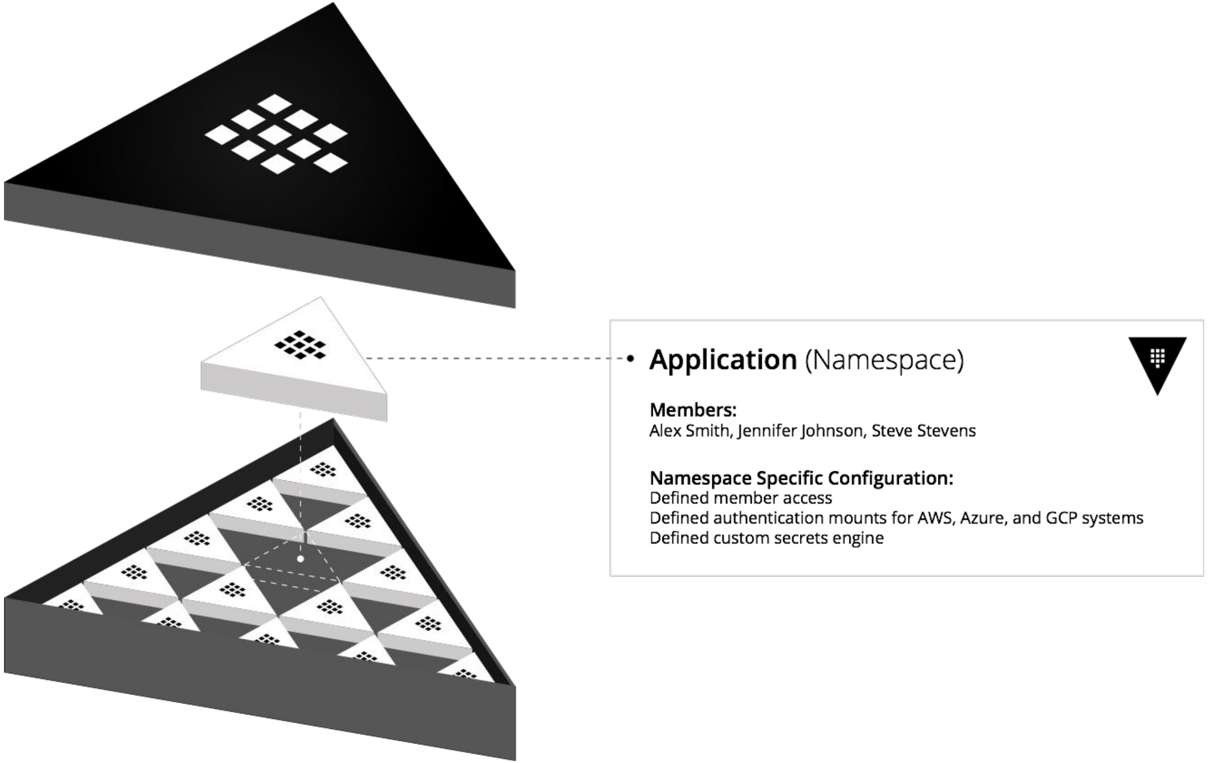
Engineering Org (Namespace)

Members:
Security Team, Operations Teams, Engineering Manager

Namespace Specific Configuration:
Defined engineering member access
Defined engineering authentication mounts
Defined engineering secrets engines

Note: Vault supports namespaces within namespaces. By default there can always be a parent managed namespace that has rights to sub-namespaces, such as the Applications/User namespaces

Namespace per User or Application



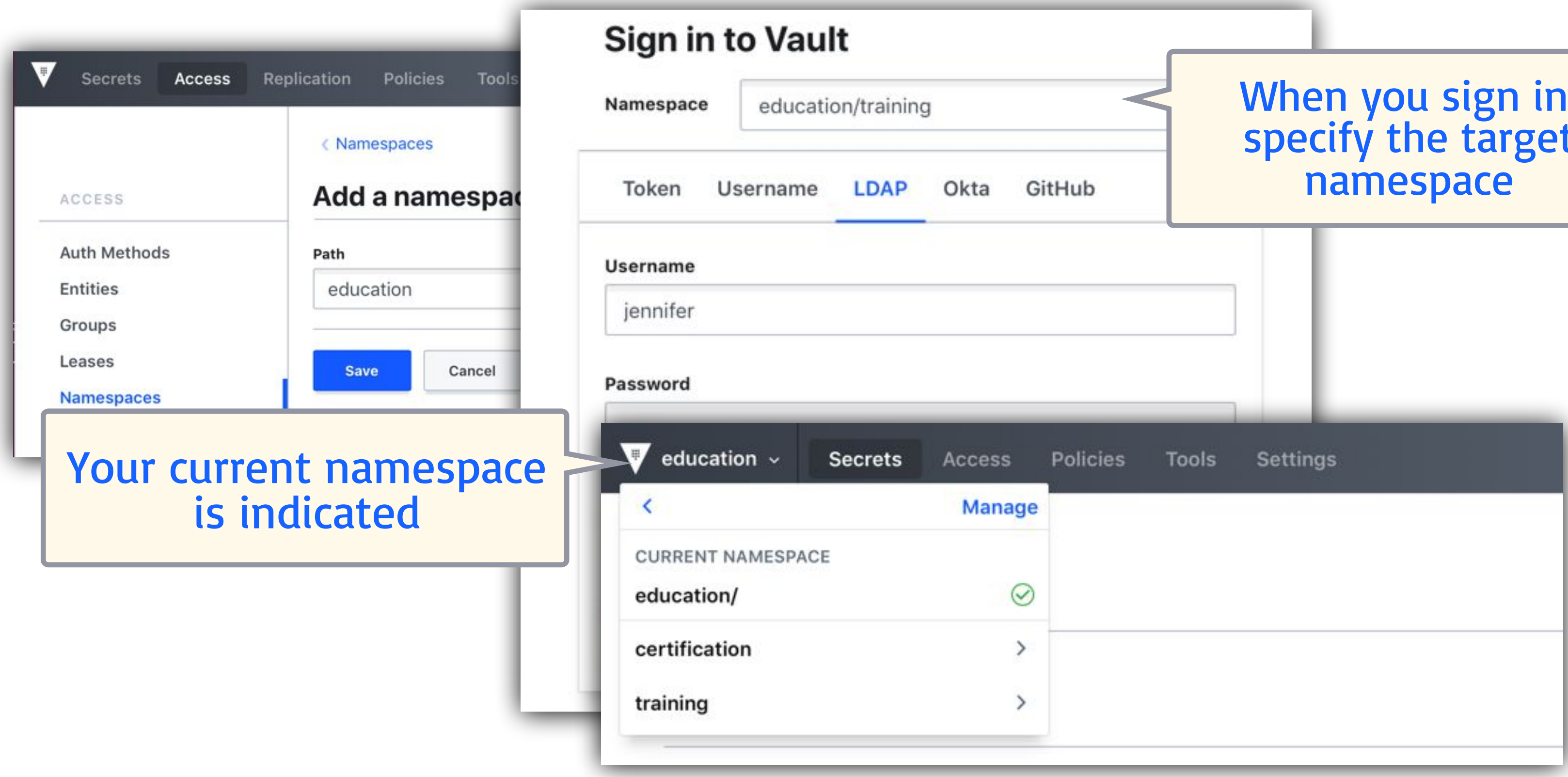
- **Application** (Namespace)

Members:
Alex Smith, Jennifer Johnson, Steve Stevens

Namespace Specific Configuration:
Defined member access
Defined authentication mounts for AWS, Azure, and GCP systems
Defined custom secrets engine

Working with Namespaces

Web UI



Add a namespac

Path: education

Sign in to Vault

Namespace: education/training

Username: jennifer

Password

CURRENT NAMESPACE

education/ ✓

certification ›

training ›

When you sign in, specify the target namespace

Your current namespace is indicated

Working with Namespaces

CLI

- To target a specific namespace in CLI command:
  - Use `-namespace` flag

```
$ vault policy write -namespace=<namespace> <policy_name> <policy_file>
```

Note: You can use -ns as a shortcut for -namespace

  - Or, set **VAULT_NAMESPACE** environment variable

```
$ export VAULT_NAMESPACE=<namespace>

$ vault policy write <policy_name> <policy_file>
```

Working with Namespaces

API

- To invoke an API on a specific namespace:
  - Pass the target namespace in the **X-Vault-Namespace** header

```
$ curl --header "X-Vault-Token: ..." \
       --header "X-Vault-Namespace: <namespace>" \
       --request GET \
       https://127.0.0.1:8200/v1/sys/mounts
```

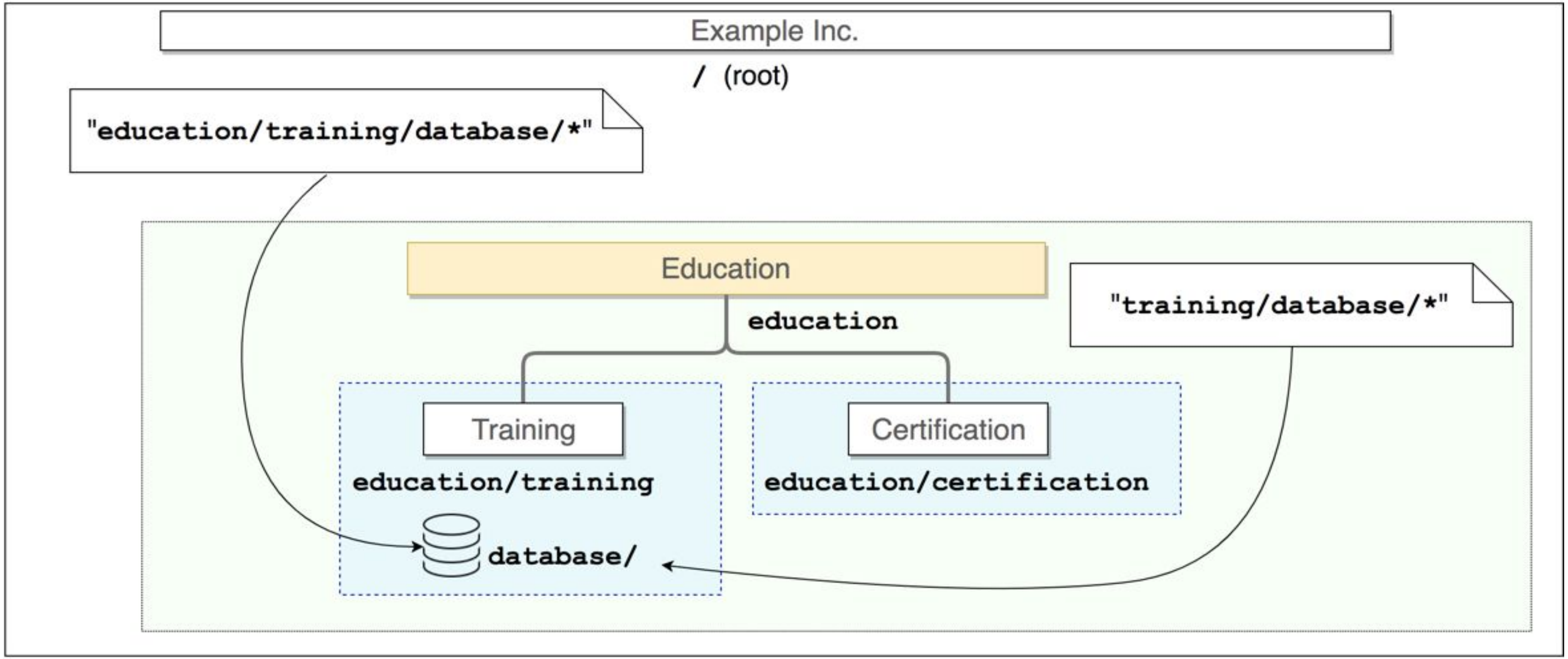
  - Or, make the namespace as a part of the API endpoint:

```
$ curl --header "X-Vault-Token: ..." \
       --request GET \
       https://127.0.0.1:8200/v1/<namespace>/sys/mounts
```

Policies

Paths

- The **policy paths** is relative to the namespace which the policy is deployed



Policy Examples

- To create a policy in the **education** namespace to give a full permission on the **education/training** namespace:

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

- To deploy a policy in the education namespace:

```
$ vault policy write -namespace=education \
          training_admin ./training_admin.hcl
```

Policy Delegation

Scenario Discussion #1

- Auth methods are enabled in each namespace

- Tokens are created in each namespace
  - Tokens created in **education** namespace are **not** valid to operate in **finance** namespace or **education/training** namespaces

- Policies are created in each namespace

Q: Bob is a *superuser* who normally operates in the **education** namespace. However, in some situation, he may need to operate in the **education/training** namespace as well. How can we accomplish this?

Solutions

- **Solution 1:** Create a policy in the **education** namespace permitting to operate in the **education/training** namespace.

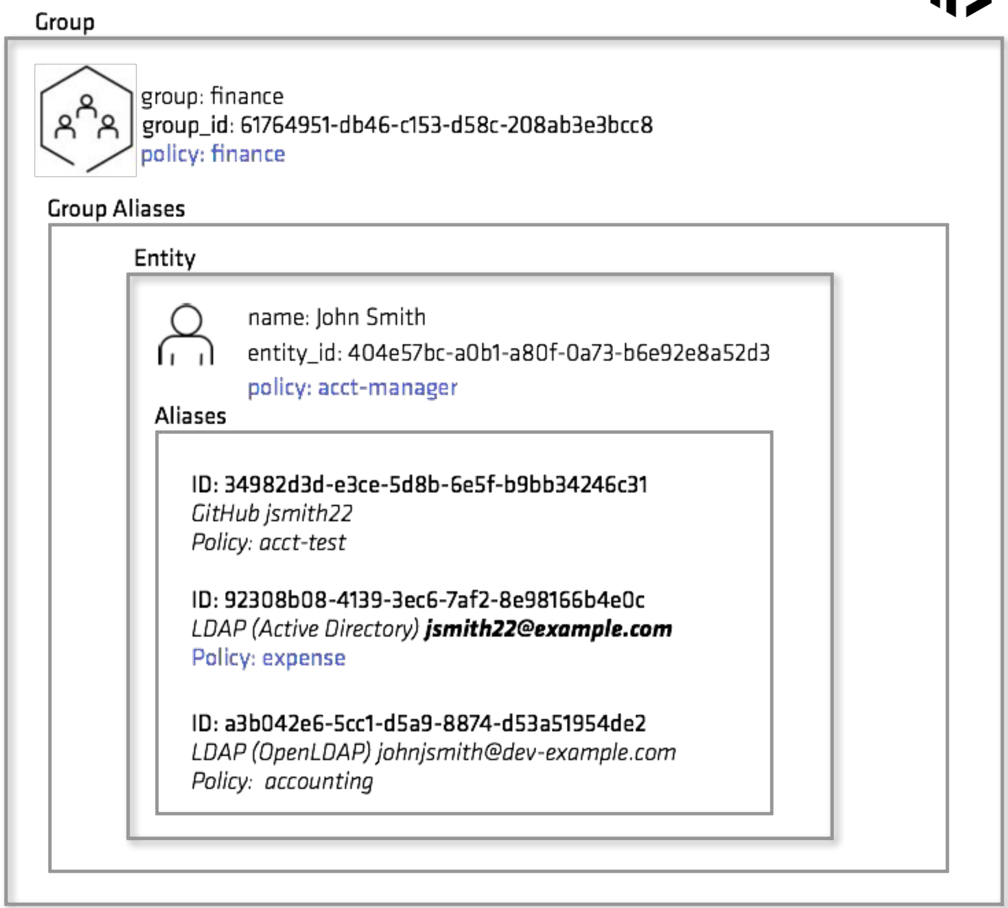
```
path "training/*" {
      capabilities = ["create", "read", "update", "delete", "list",
"sudo"]
}
```

- **Solution 2:** Since identity groups can pull in entities and groups from other namespaces, add Bob's entity to the identity group in the **education/training** namespace.

Quick Review!

Identity

- Tokens inherits policies from identities
  - Identity entity, John Smith has **acct-manager** policy attached
  - Identity group, finance has **finance** policy attached
  - Member alias inherits both policies



Group

group: finance
group_id: 61764951-db46-c153-d58c-208ab3e3bcc8
policy: finance

Group Aliases

Entity

name: John Smith
entity_id: 404e57bc-a0b1-a80f-0a73-b6e92e8a52d3
policy: acct-manager

Aliases

ID: 34982d3d-e3ce-5d8b-6e5f-b9bb34246c31
GitHub jsmith22
Policy: acct-test

ID: 92308b08-4139-3ec6-7af2-8e98166b4e0c
LDAP (Active Directory) ***jsmith22@example.com***
Policy: expense

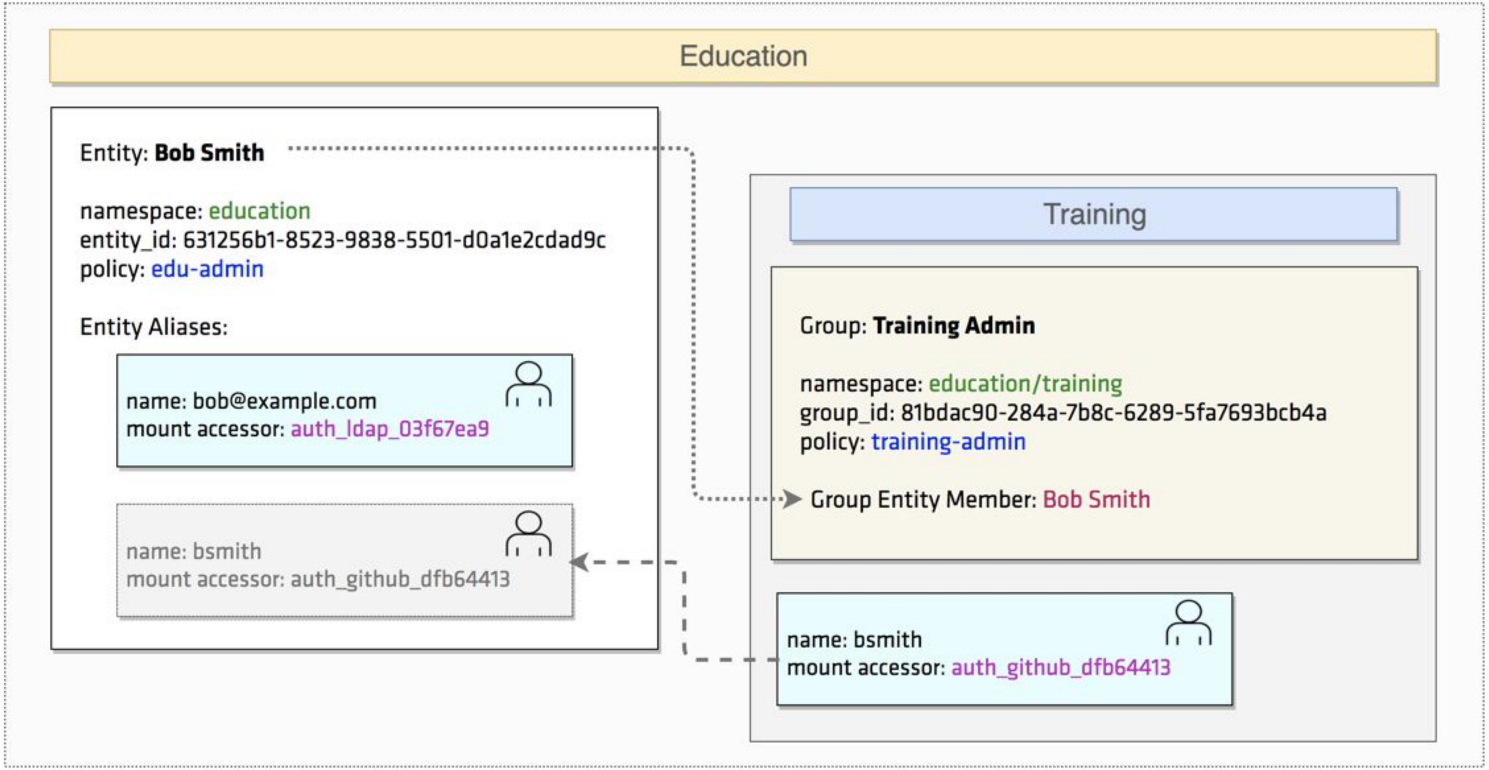
ID: a3b042e6-5cc1-d5a9-8874-d53a51954de2
LDAP (OpenLDAP) johnjsmith@dev-example.com
Policy: accounting

Use Identity Entity and Group

1. Enable an auth method in each namespace so that users can authenticate
   - e.g. education/auth/ldap
   - e.g. education/training/auth/github
2. Create an identity entity in **education** namespace to tie in Bob's user accounts in two namespaces
3. Create an identity group in **education/training** namespace and add Bob's identity entity as a group member

Bob will inherits policies that are attached to the entity and group.

Bob’s Case



Education

Entity: **Bob Smith**

namespace: education
entity_id: 631256b1-8523-9838-5501-d0a1e2cdad9c
policy: edu-admin

Entity Aliases:

name: bob@example.com
mount accessor: auth_ldap_03f67ea9

name: bsmith
mount accessor: auth_github_dfb64413

Training

Group: **Training Admin**

namespace: education/training
group_id: 81bdac90-284a-7b8c-6289-5fa7693bcb4a
policy: training-admin

Group Entity Member: Bob Smith

name: bsmith
mount accessor: auth_github_dfb64413

Namespaced Policies vs. Identity

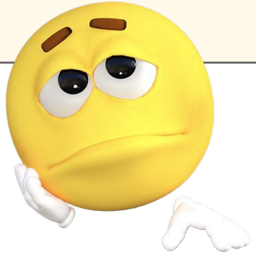
Solution 1: Policy paths with Namespace	Solution 2: Identity Entities & Groups
● Bob cannot authenticate against the **education/training** namespace using the auth method configured in the **education** namespace ● Bob cannot log into the **education/training** namespace using the token created in the **education** namespace ● The training-admin policy exists in the **education** namespace; therefore, it cannot be assigned to users in the **education/training** namespace	● An auth method must be enabled and configured in each namespace ● Bob must use the auth method enabled in the respective namespace to authenticate ● Bob must use the token specific to each namespace ● The policy is reusable ○ Deploy the same policy in other namespaces ○ Attach the policy to other users in the **education/training** namespace

Auth Method Propogation

Scenario Discussion #2

- Example Inc. has a company-wide **LDAP server** where employee's group memberships are defined

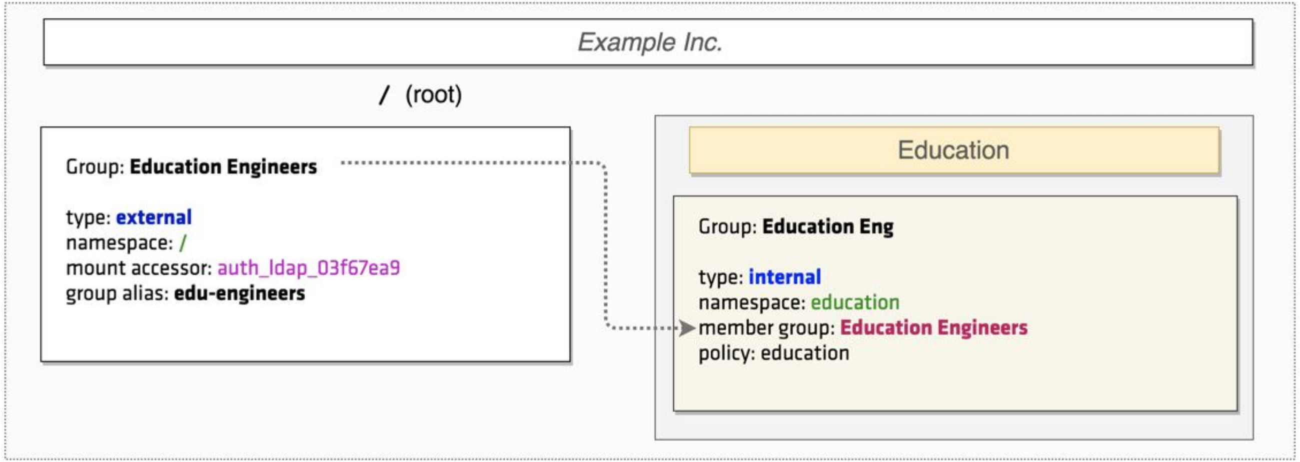
Q: I don't want to enable an LDAP auth method in every namespace! I want to enable the LDAP auth method in the **root** namespace and propagate it down to other namespaces. What do I do?



Solution

1. Enable the LDAP auth method, create and configure an **external identity group** in the **root** namespace for the LDAP group

2. Create an internal group in the **education** and/or **education/training** namespaces which has the external group as its member group



Example Inc.

/ (root)

Group: **Education Engineers**

type: **external**
namespace: /
mount accessor: auth_ldap_03f67ea9
group alias: **edu-engineers**

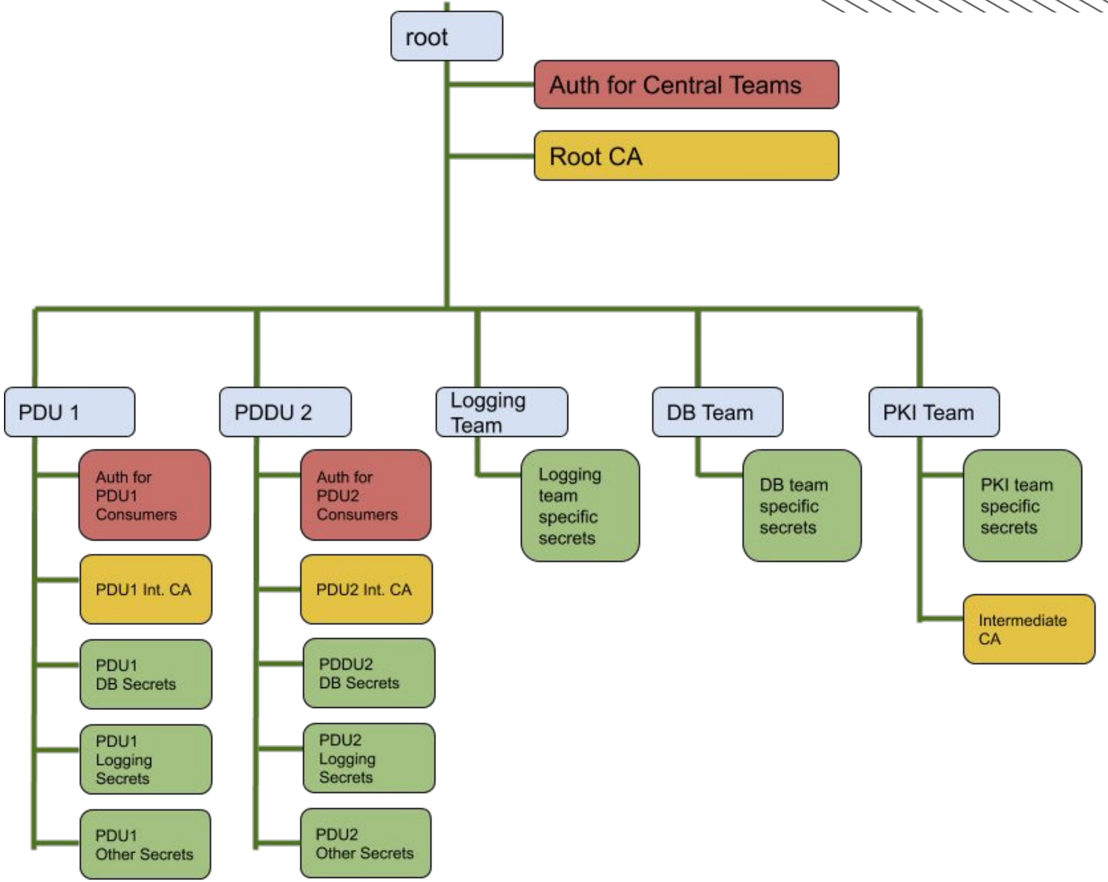
Education

Group: **Education Eng**

type: **internal**
namespace: education
member group: **Education Engineers**
policy: education

EA Daylight Robbery

Customer Example



Demo

Demo

1. Clone https://github.com/allthingsclowd/vault_ldap_namespaces
2. Follow the instructions and you'll have a Vault and LDAP playground which may be useful for customer demos
3. Apologies - this is NOT beautiful

Thank you.

HashiCorp

www.hashicorp.co
m

hello@hashicorp.co
m