Describe and Interpret Multi-Tenancy with Namespaces

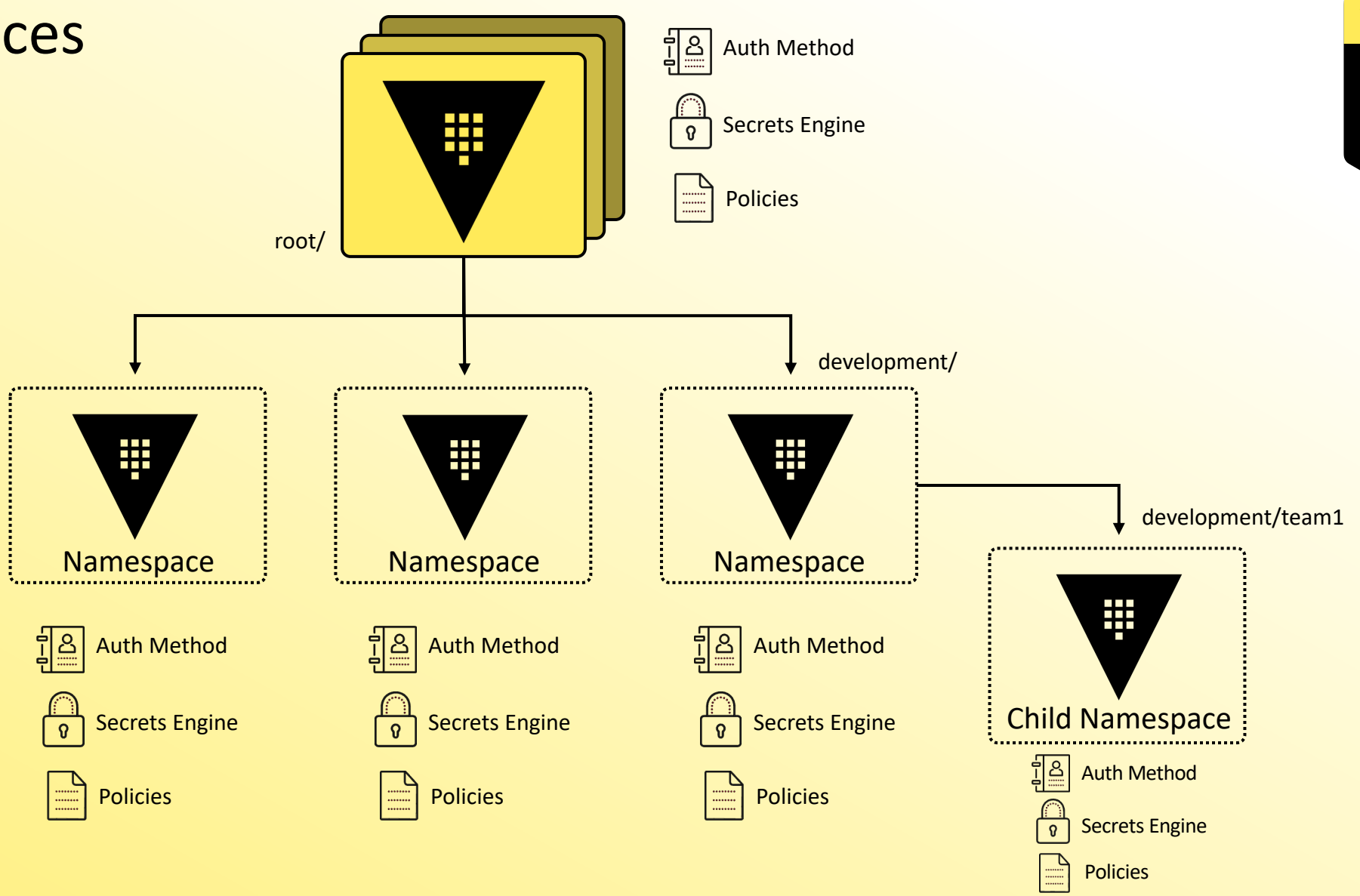
What are Namespaces?

- Allows organizations to provide “Vault as a Service”
  - Provides isolated environments on single Vault environment
  - Multi-tenant but centralized management
  - Allows delegation of Vault of responsibilities
- Available in all versions of Vault Enterprise
- Each namespace can have its own:
  - Policies
  - Auth Methods
  - Secrets Engines
  - Tokens
  - Identities

What are Namespaces?

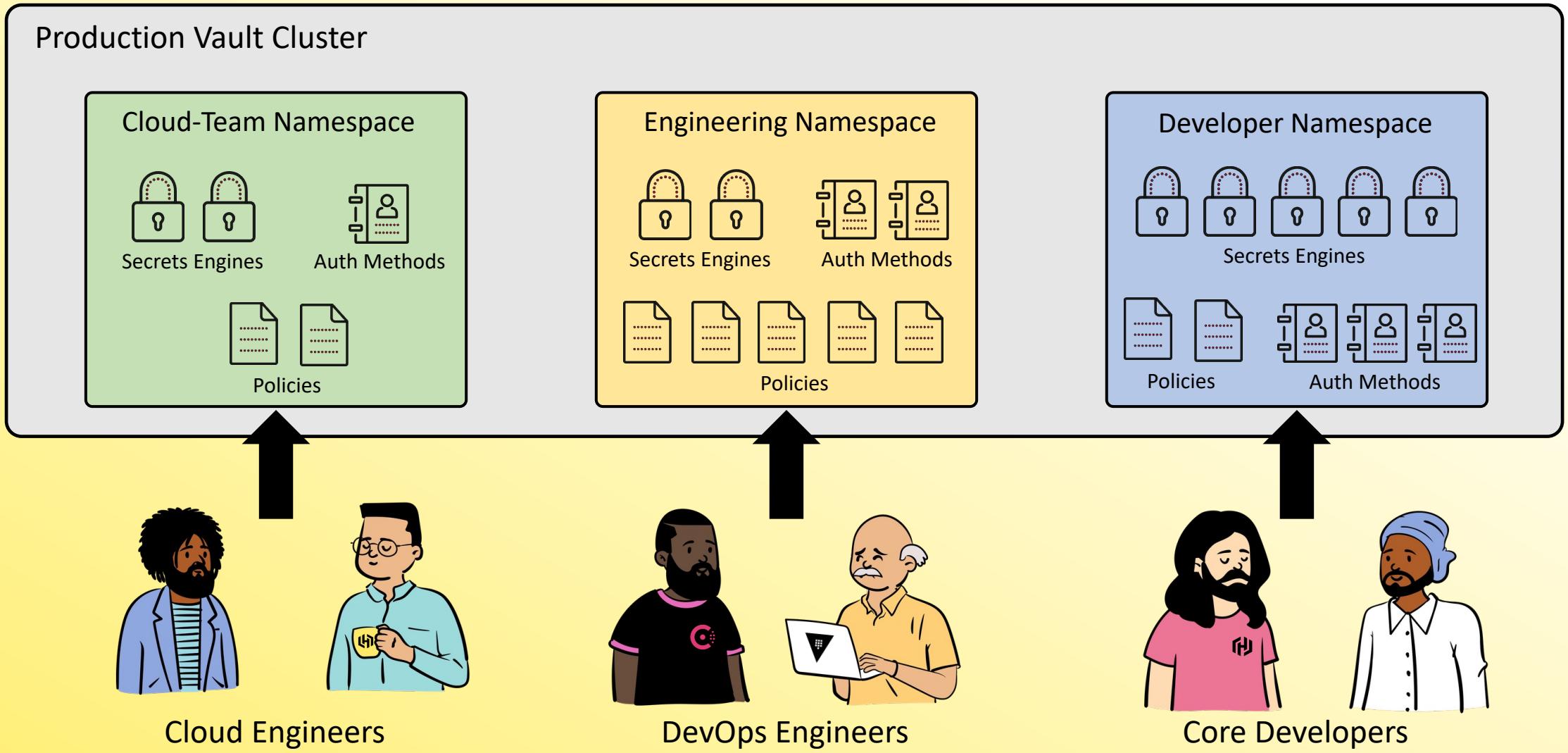
- The default namespace is ‘root’
- Namespaces are created in a hierarchical fashion
- Just like root, paths and ACLs are relative to the namespace, making easier to re-use commands and policies across multiple namespaces
- Tokens are only valid in a single namespace, but you can create an entity who has access to other namespaces

Namespaces

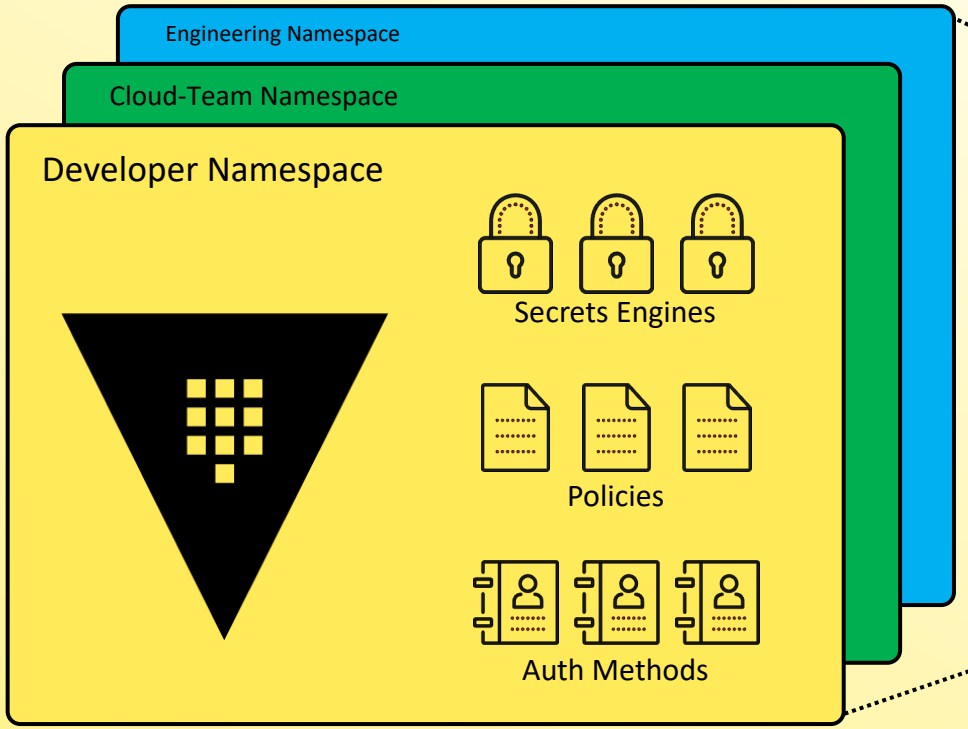

root/
Auth Method
Secrets Engine
Policies
development/
development/team1
Namespace
Namespace
Namespace
Auth Method
Secrets Engine
Policies
Auth Method
Secrets Engine
Policies
Auth Method
Secrets Engine
Policies
Child Namespace
Auth Method
Secrets Engine
Policies


Vault
CERTIFIED
OPERATIONS
PROFESSIONAL

Assigning Namespaces



Administrative Delegation



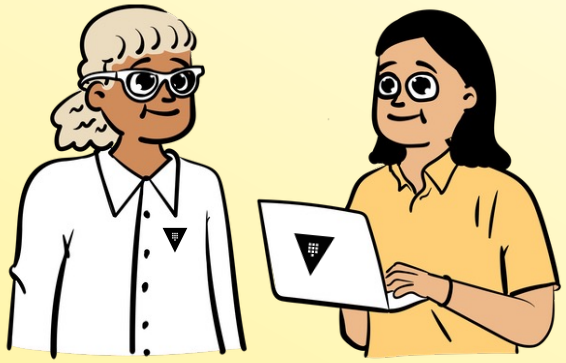
Vault Cluster

Developer Namespace Admin

Responsible for Dev:

- Secrets Engines
- Policies
- Auth Methods

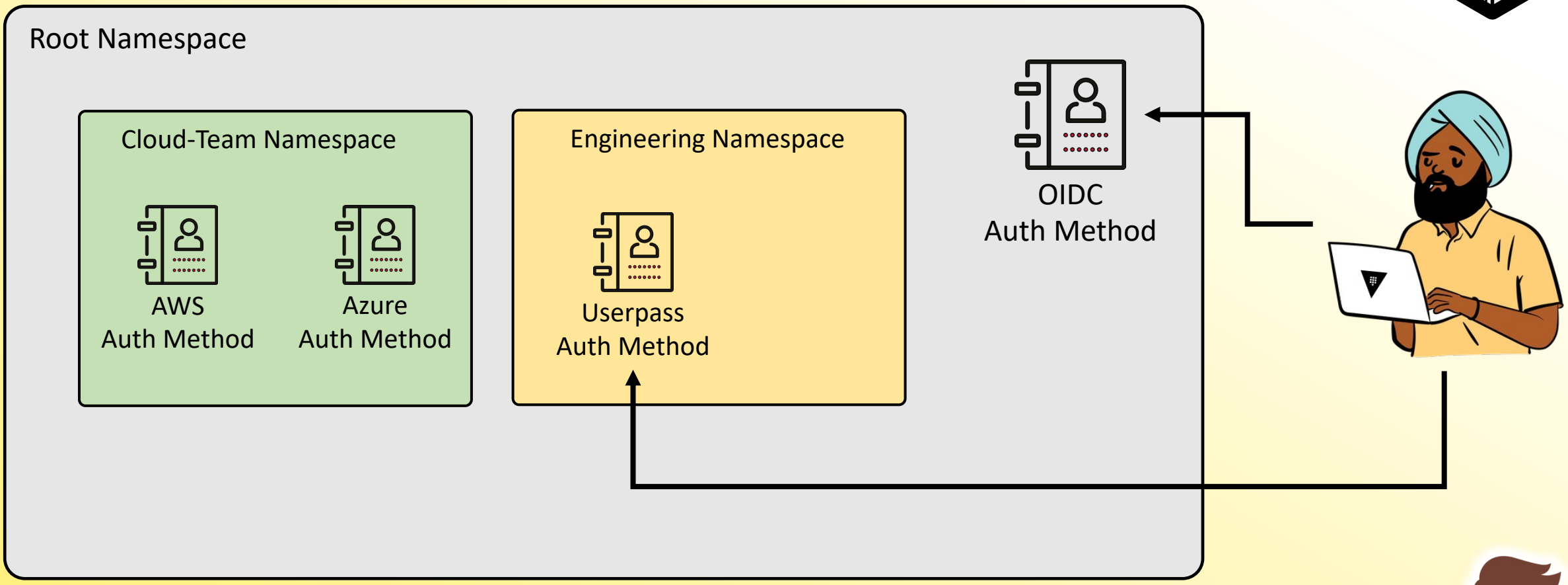
Vault Engineers



Responsible for:

- Cluster Nodes
- Audit Devices
- Root Namespace
- Storage Backend
- Vault Upgrades

Authenticating to Namespaces



Common Namespace Commands

Create Namespace

```
$  vault namespace create <namespace>
```

List Namespaces

```
$  vault namespace list
```

Delete a Namespace

```
$  vault namespace delete <namespace>
```

Using Namespaces on the CLI

Set Namespace Environment Variable – then run commands as normal

```
$  export VAULT_NAMESPACE=<namespace>
```

Reference a Namespace on the CLI when running a command

```
$  vault kv get -namespace=<namespace> kv/data/sql/prod
```

Referencing Namespaces in the API

Add the API Header = `X-Vault-Namespace`

```
curl \
 -header "X-Vault-Token: "hvs.a83b50ed2aa548212" \
 -header "X-Vault-Namespace: "development/" \
 -request GET \
 https://vault.hcvop.com:8200/v1/kv/data/sql/prod
```

Referencing Namespaces in the API

Add the Namespace to the API Endpoint

```
curl \
 -header "X-Vault-Token: "hvs.CAESIA7Y-LwSxnE926onQwdxIUF7" \
 -request GET \
 https://vault.hcvop.com:8200/v1/development/kv/data/sql/prod
```

Writing Policies for Namespaces

The path is relative to the Namespace

Root Namespace

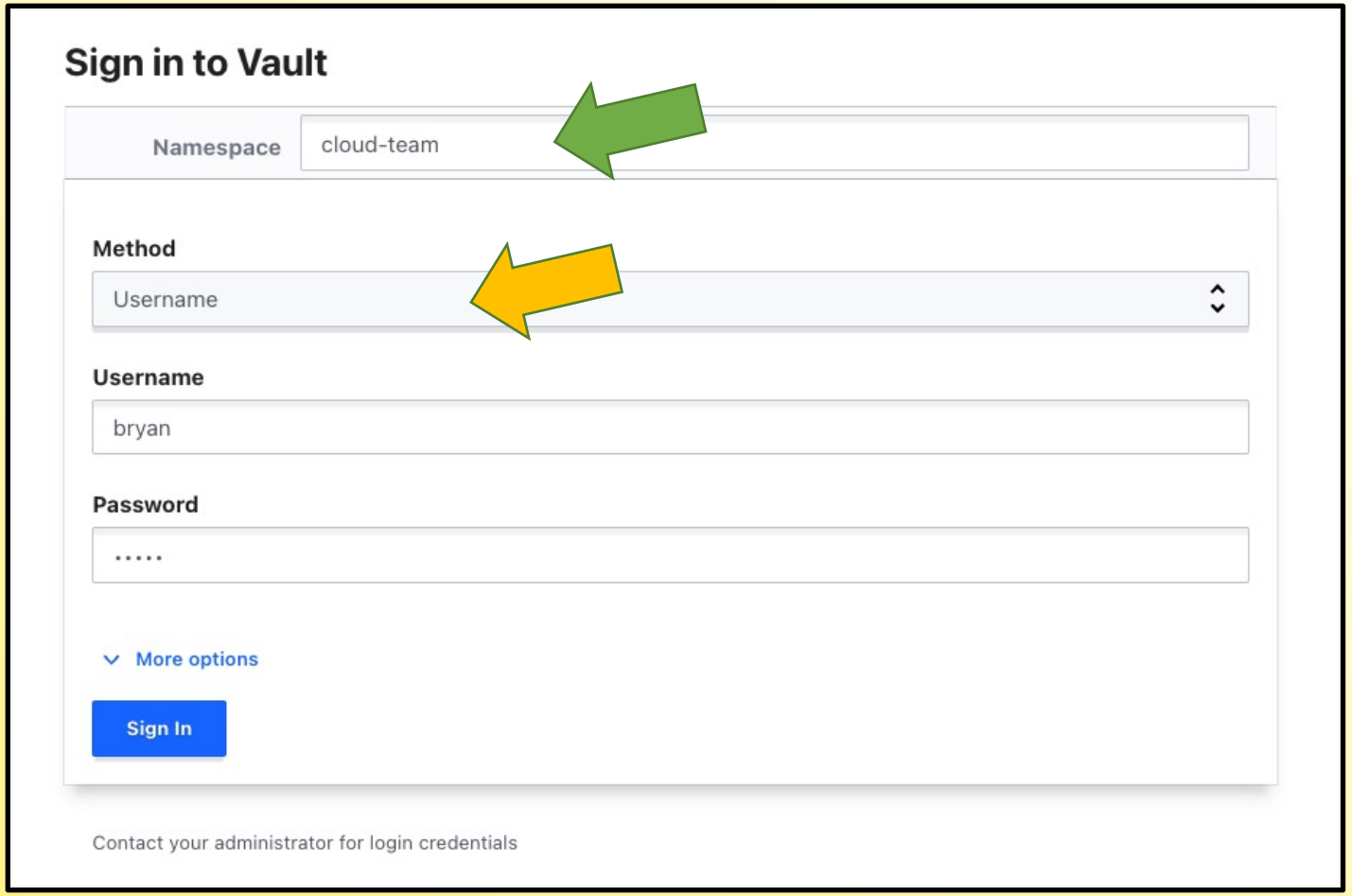
Cloud-Team Namespace

database/

```
path = "database/creds/prod-db" {
  capabilities = ["read"]
}
```

```
path = "cloud-team/database/creds/prod-db" {
  capabilities = ["read"]
}
```

Authenticating to a Namespace via UI



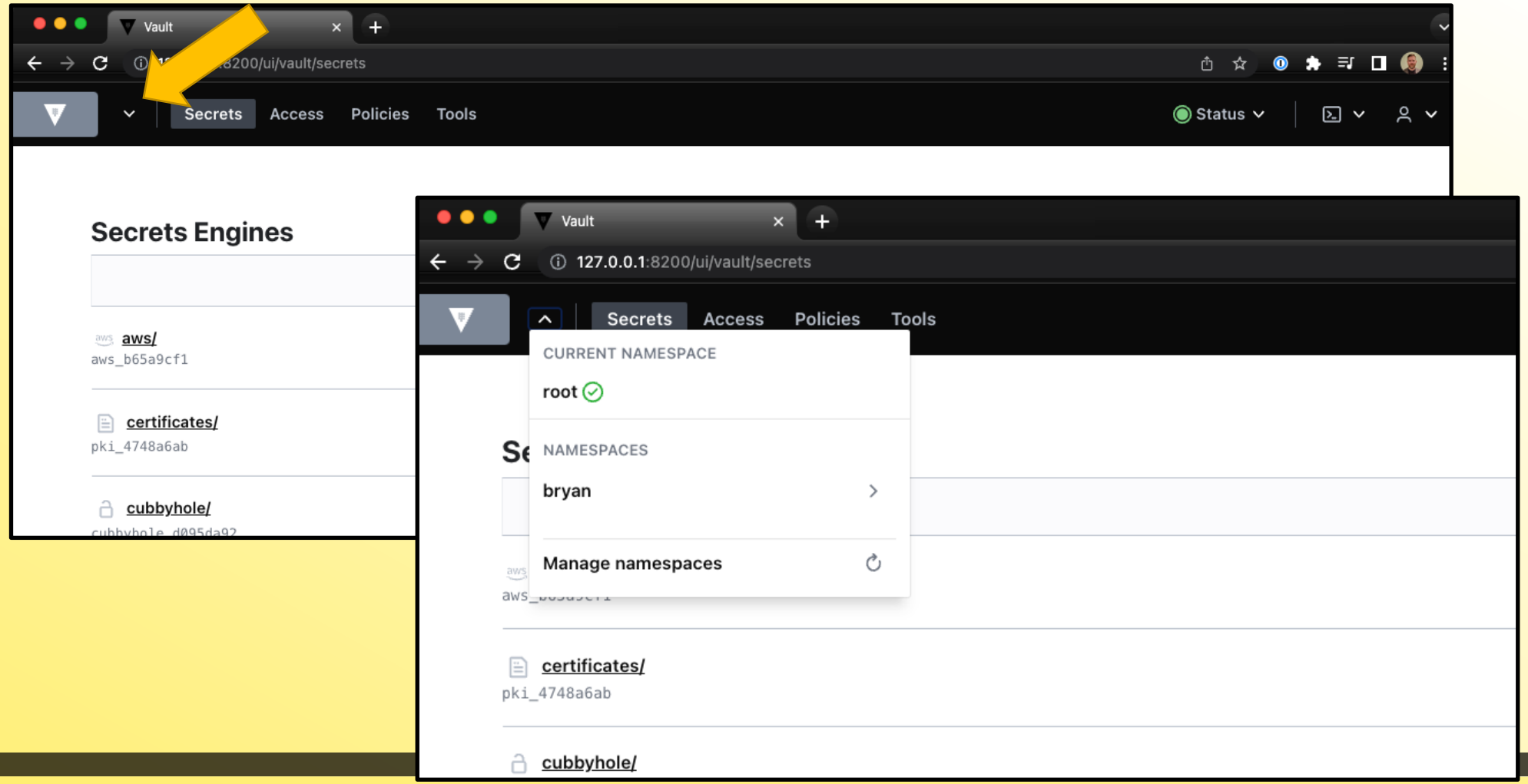
Authenticating to a Namespace via CLI


```
$ vault login -namespace=cloud-team -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
hvs.CAESIM5RikdMODs5nZrFrsecgqUKggrnXgSOZrkvXMtUXnwKGicKImh2cy5oOXlrNWFQRHNQM1Y4M
G5xZkF0VFB6dVcubjU3eTYQwAM
token_accessor         rOH7HYtHmZ6fDX4z0RCJVxbF.n57y6
token_duration         768h
token_renewable        true
token_policies         ["default"]
identity_policies      []
policies               ["default"]
token_meta_username    bryan
```

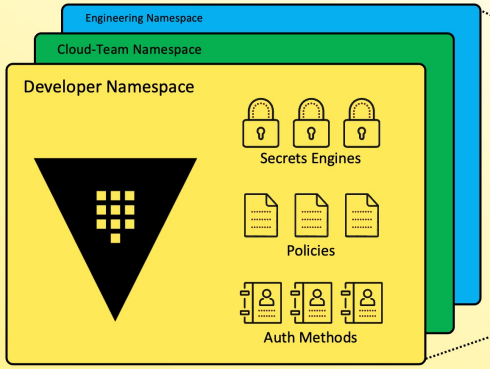
Enabling an Auth Method In a Namespace


```
$ vault namespace create cloud-team
Key     Value
---     -----
id      n57y6
path    cloud-team/

# Enable userpass auth method using the namespace flag
$ vault auth enable -namespace=cloud-team userpass
Success! Enabled userpass auth method at: userpass


# Enable aws auth method using environment variable
$ export VAULT_NAMESPACE=cloud-team
$ vault auth enable aws
```

Working with Namespaces in the UI



Administrative Delegation



Vault Cluster

Developer Namespace Admin

Responsible for Dev:
- Secrets Engines
- Policies
- Auth Methods

Vault Engineers

Responsible for:
- Cluster Nodes
- Audit Devices
- Root Namespace
- Storage Backend
- Vault Upgrades

Control Group Workflow

Here's my accessor. Please approve.
cqL9n3r4kMeIQZekoLrMWMWN

1. GET kv/data/customers/orders

2. Response

```
"wrap_info": {
    "token": "hvs.CAESIPvNkRgluUVNT_ccLsm6aZ-",
    "accessor": "cqL9n3r4kMeIQZekoLrMWMWN",
    "ttl": 300,
    ...
```

Policy with Group Control on kv/data/customers/orders

Account Managers

3. Share Accessor with Managers for Approval

Assigning Namespaces

Production Vault Cluster

Cloud-Team Namespace
Secrets Engines
Auth Methods
Policies

Engineering Namespace
Secrets Engines
Auth Methods
Policies

Developer Namespace
Secrets Engines
Policies
Auth Methods

Cloud Engineers

DevOps Engineers

Core Developers

Oh No...Our Cluster...It's Broken

Data Center A

Primary Cluster

DR Replication Cluster

Data Center B

Performance Replication Cluster

DR Replication Cluster