Use Batch Tokens

Introduction to Batch Tokens

Batch tokens are encrypted binary large objects (blobs)

- Designed to be lightweight & scalable
- They are NOT persisted to storage, but they are not fully-featured
- Ideal for high-volume operations
- Can be used for DR Replication cluster promotion as well
- Includes information such as policy, TTL, and other attributes

- Batch tokens are encrypted using the barrier key, which is why they can be used across all clusters within the replica set

Compare Batch Tokens vs. Service Tokens

	Service Tokens	Batch Tokens
Can be root tokens	Yes	No
Can create child tokens	Yes	No
Renewable	Yes	No
Listable	Yes	No
Manually Revocable	Yes	No
Can be periodic	Yes	No
Can have explicit Max TTL	Yes	No (always uses a fixed TTL)
Has accessors	Yes	No
Has Cubbyhole	Yes	No
Revoked with parent (if not orphan)	Yes	Stops Working
Dynamic secrets lease assignment	Self	Parent (if not orphan)

Know these differences very well

Compare Batch Tokens vs. Service Tokens

	Service Tokens	Batch Tokens
Can be used across Performance Replication clusters	No	Yes (if orphan)
Creation scales with performance standby node count	No	Yes
Cost	Heavyweight; multiple storage writes per token creation	Lightweight; no storage cost for token creation

Know these differences very well

Identifying Token Types in Vault via Prefix

Token Type	Vault 1.9.x or earlier	Vault 1.10 and later
Service tokens	s.	hvs.
Batch tokens	b.	hvb.
Recovery tokens	r.	hvr.

Identifying Token Types in Vault via Prefix

hvs.

Service Token

hvb.

Batch Token

hvr.

Recovery Token

Token Size

Service Token ⟶

Size: ~98 bytes

hvs.CAESIA4CZQisJNn9eq3g5TS5xP0-
DPkFDsshIi_jb5UH28AuGiAKHGh2cy5wZjl
PU1NsVlpWaTQxSFUyczFuQk9DOFgQHQ

Batch Token ⟶

Size: ~128 bytes

hvb.AAAAAQKskxnAqTz0Ah3qu5Hc4Q3lY
dqCocdDZjLXhyLAjuhhBJktOCrBaIJVbKwE
6AVSxD6WAFvI2ZUHs2MUb1gcpqYvro-
kfVv10x7tKZ9GqUObUwKnn5341sU-

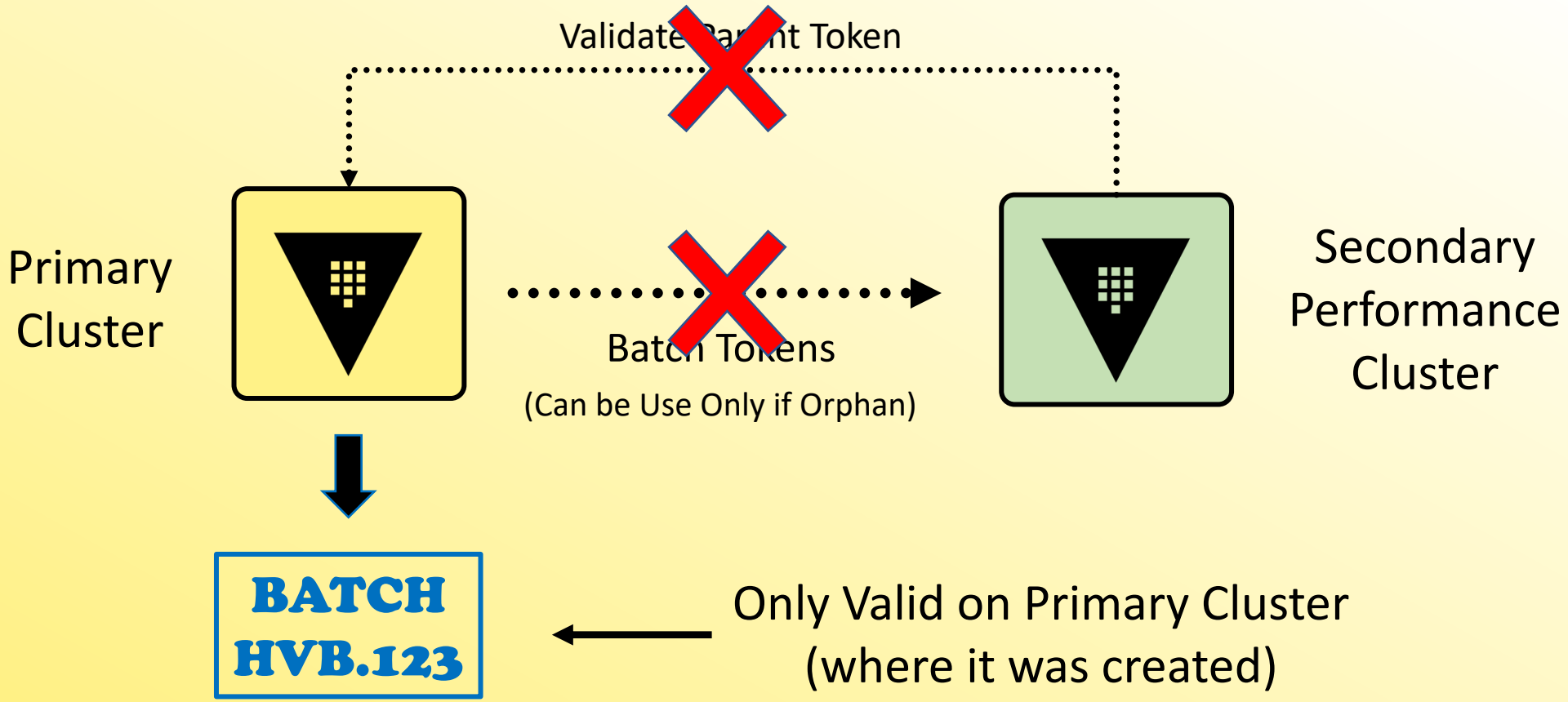
Token Size

Initial Root Token ⟶ hvs.JTjQKbLZOja5LO2anRbGjG6h

Size: ~28 bytes

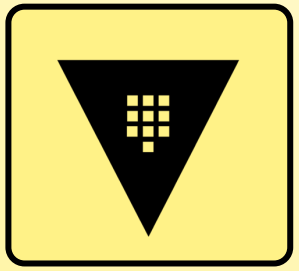
- Token sizes can change. In Vault 1.10, they changed it to 95+ bytes.
- HashiCorp recommends that you plan for a maximum length of 255 bytes to future proof yourself if you have workflows that rely on the token size

Using Batch Tokens – Non-Orphan Token

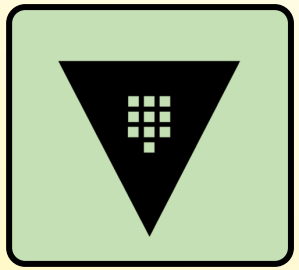


Using Batch Tokens – Orphaned Token

Primary Cluster



Batch Tokens
(Can Be Used Only if Orphan)



Secondary Performance Cluster

BATCH
HVB.987
(orphan)

Has no parent, so we don't need to validate parent
(Valid on any cluster in replica set)

Creating a Batch Token

Available to Use Across Perf Clusters

Terminal

```
$ vault token create -type=batch -orphan=true -policy=hcvop
Key                  Value
---                  -----
Token                hvb.AAAAAQKskxnAqTz0Ah3qu5Hc4Q3lYdqCocdDZ
jLXhyLAjuhhBJktOCrBaIJVbKwE6AVSxD6WAFvI2ZUHs2MUb1gcpqYvro-kfVv
10x7tKZ9GqUObUwKnn5341sU-
token_accessor       n/a
token_duration       768h
token_renewable      false
token_policies       ["default" "hcvop"]
identity_policies    []
policies             ["default" "hcvop"]
```

Creating a Batch Token

Terminal

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

DR Operation Batch Token

- *As presented in previous objective, y*ou can use a batch token to promote a DR secondary cluster
    - Eliminates the requirement to generate a DR operation token using the unseal/recovery keys

- This can be a strategic operation that the Vault Operator can to do prepare for an unexpected loss of the primary cluster
- However, the batch token must have the proper permissions to promote a secondary and perform related actions

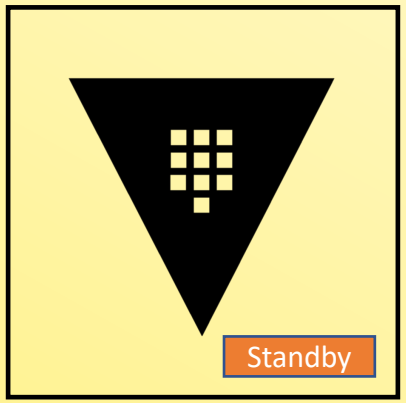
DR Operation Batch Token

```
path "sys/replication/dr/secondary/promote" {
  capabilities = ["update"]
}

# To update the primary to connect
path "sys/replication/dr/secondary/update-primary" {
  capabilities = ["update"]
}

# Only if using integrated storage (raft) as the storage backend
# To read the current autopilot status
path "sys/storage/raft/autopilot/state" {
  capabilities = ["update" , "read"]
}
```

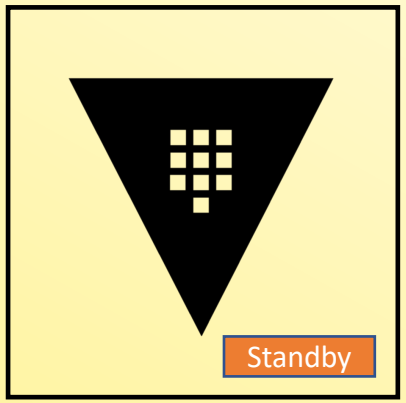
Describe the Use Cases of Performance Standby Nodes

Vault Clustering - OSS

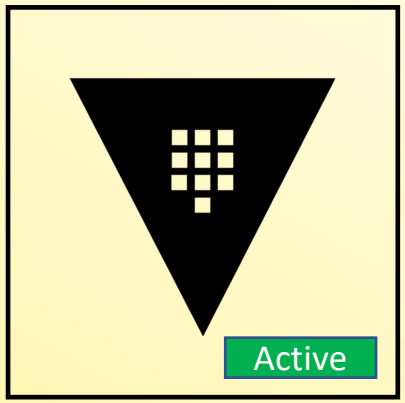
Vault Node A



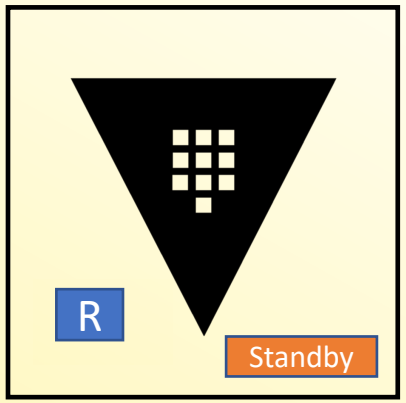
Vault Node B



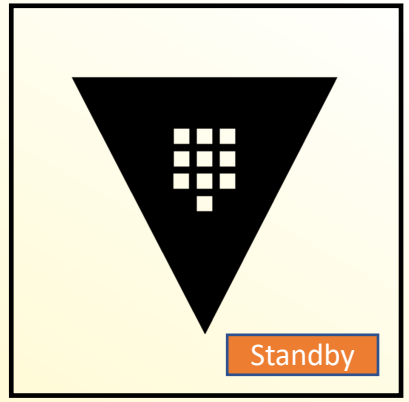
Vault Node C



Vault Node D



Vault Node E



√ read
√ write

X read
X write

Developer

~~Credential Request~~

Credential Request

Vault OSS is a scale UP application

Vault Clustering - Enterprise

Vault Node A

Performance Standby

√ read
X write

Vault Node B

Performance Standby

√ read
X write

Vault Node C

Active

√ read
√ write

Only Forward Write Request

Vault Node D

R

Performance Standby

√ read
X write

Read Response

Vault Node E

Performance Standby

√ read
X write

√ read
√ write

Developer

~~Credential Request~~

Credential Request

Vault Enterprise is a scale OUT application

What is a Read?

Any operation that does **NOT** result in a storage write is considered a **READ**

- Not necessarily limited to **HTTP GET** or `vault read` operations

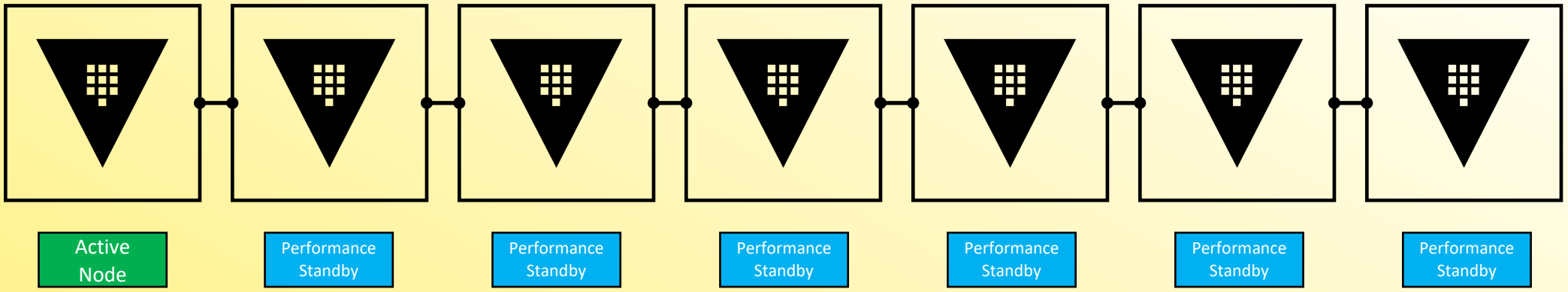
- Common read-only actions performed by applications may include:
    1. Reading secrets stored in the Key/Value engine
    2. Transit Secrets Engine – Encrypt or Decrypt operations
    3. Sign SSH client keys

Vault Enterprise with Performance Standby

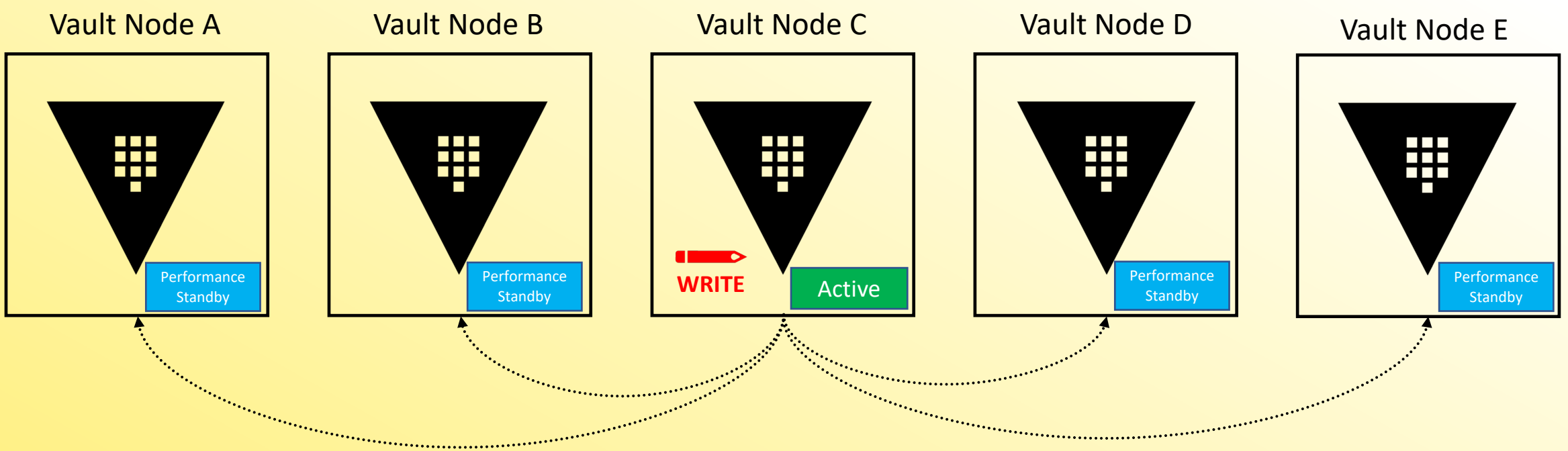
- To scale a Vault Enterprise cluster, performance standby nodes can respond to read requests from clients rather than sending the request to the Active node
- Applications known to require reads can be directed to performance standby nodes
    - this will help offload traffic from the Active node and allow you to scale **OUT** your cluster
- Performance Standby nodes can still take over as an Active node to continue providing high-availability within the local cluster

💡 **Reminder:** Performance Standby functionality is a Vault Enterprise feature

Scaling Out with Performance Secondaries

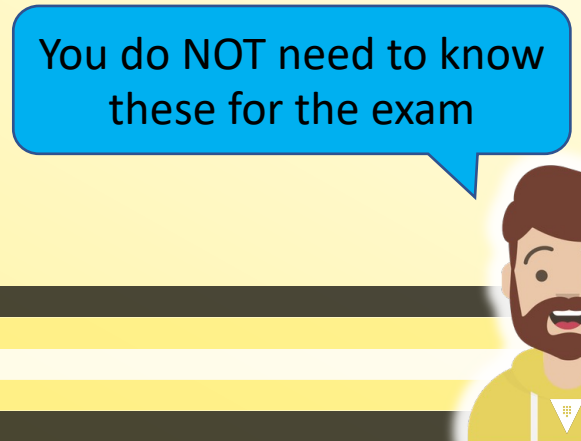


Eventual Consistency



How Do I Target a Performance Standby?

- Vault provides health information via the `/sys/health` endpoint
- Load Balancers can target specific return codes to determine an Active node vs. a Performance Standby node
- The default status codes include:
  - **`200`** – initialized, unsealed, and active node
  - **`429`** – unsealed but standby node
  - **`472`** – DR replication secondary and active node
  - **`473`** – Performance Standby
  - **`501`** – Not Initialized
  - **`503`** – Sealed node

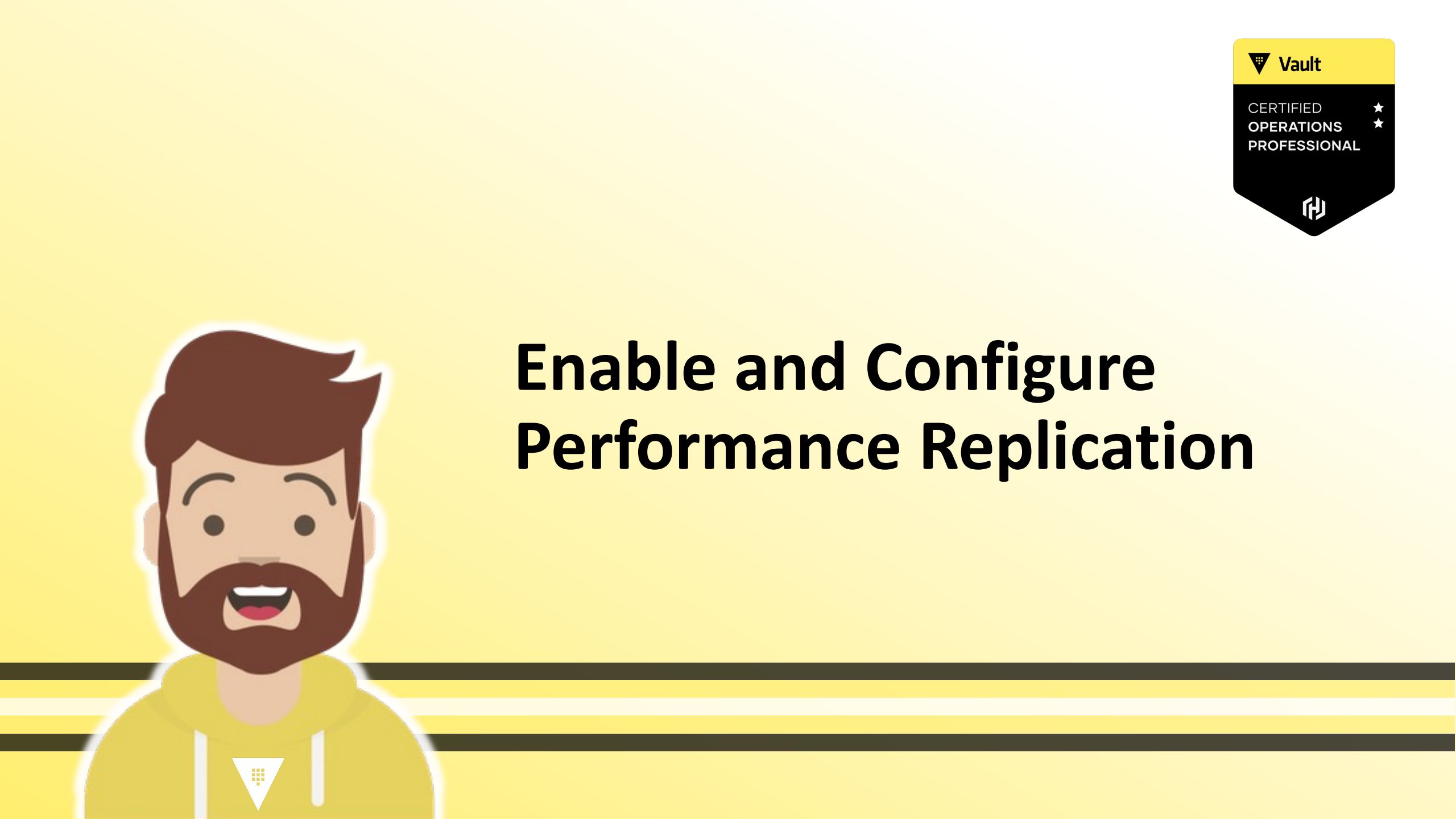


How Do I Enable Performance Standby

It's enabled by default for Vault Enterprise - if licensed

You can disable it if you want by adding the following flag:

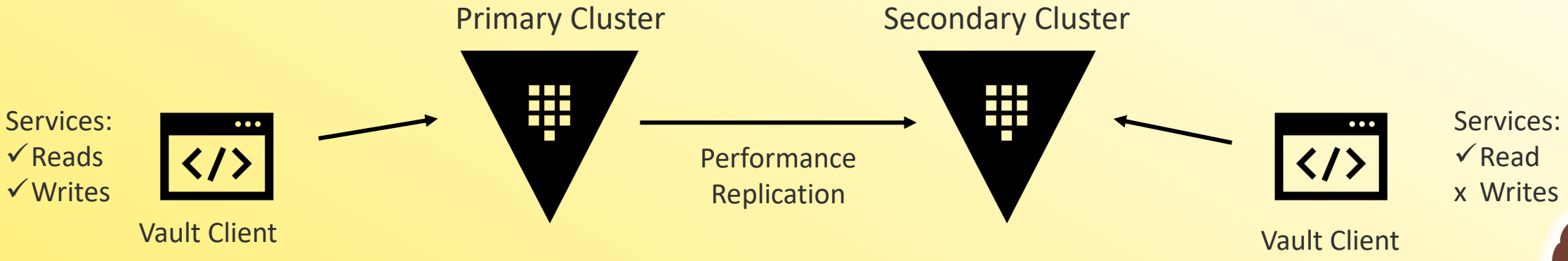
```
disable_performance_standby=true
```



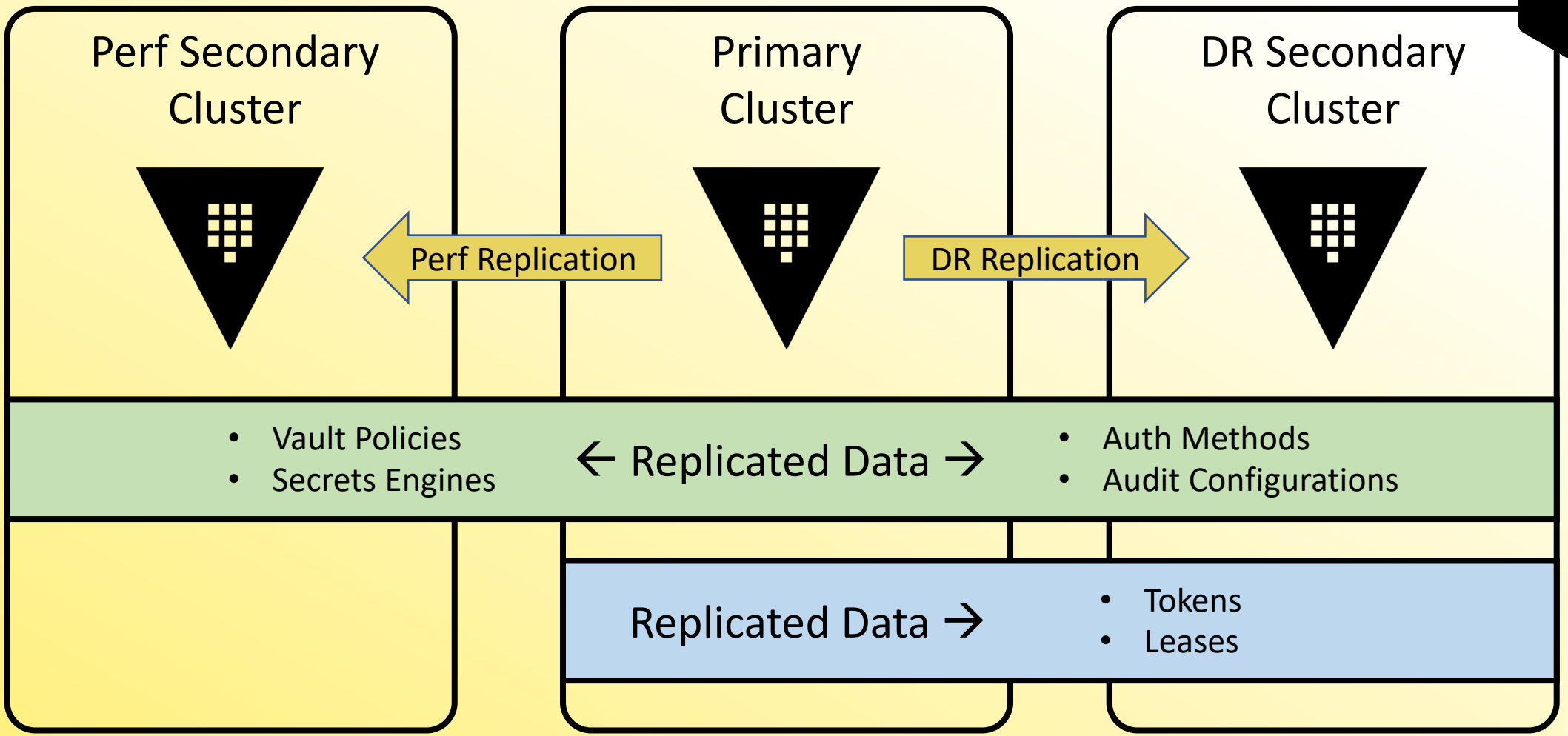
Enable and Configure Performance Replication

Intro to Performance Replicaiton

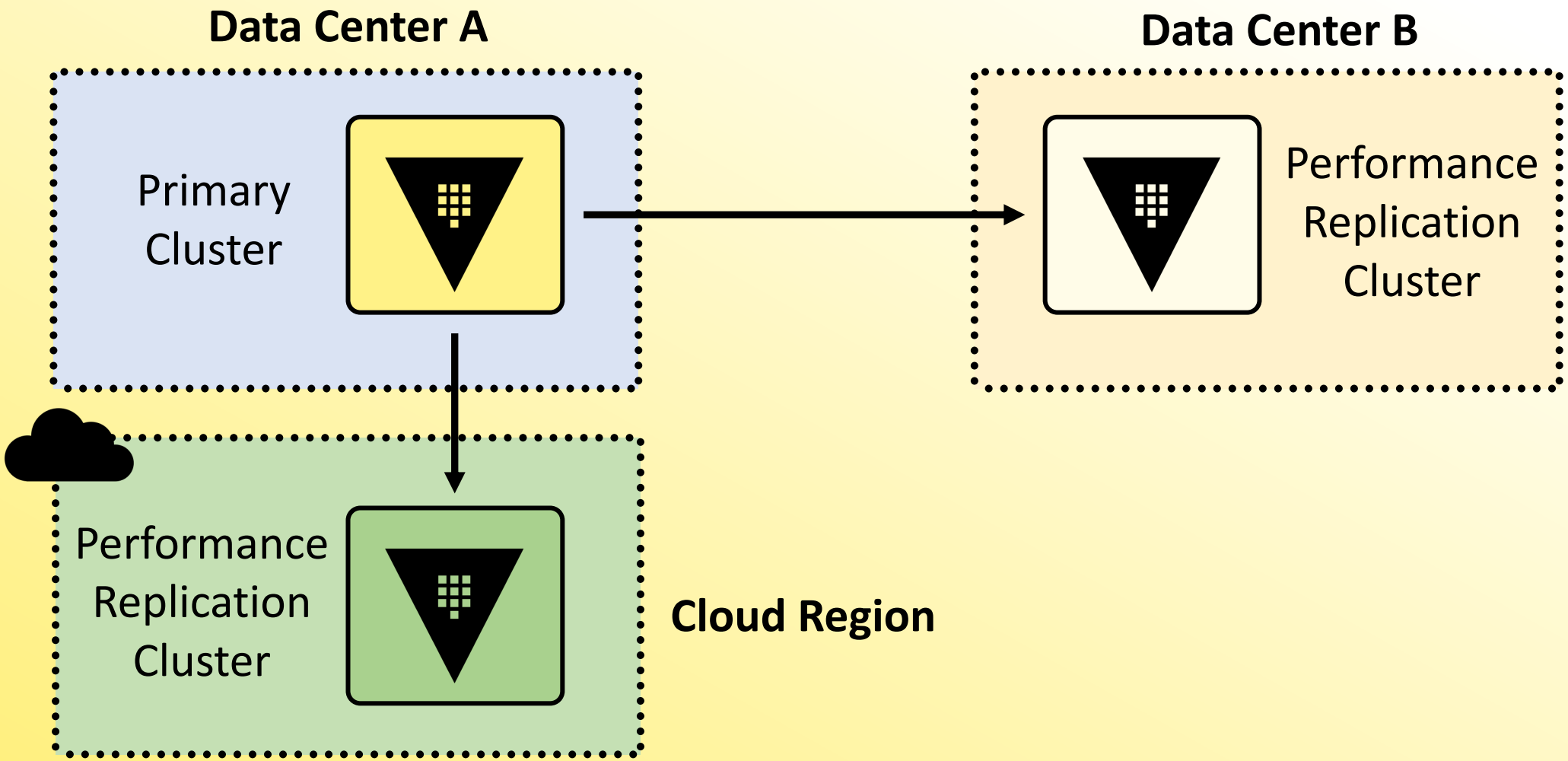
- Replicates the underlying configuration, policies, and other data
- Ability to service reads from client requests
- Clients will authenticate to the performance replicated cluster separately
- Does not replicate tokens or leases to performance secondaries



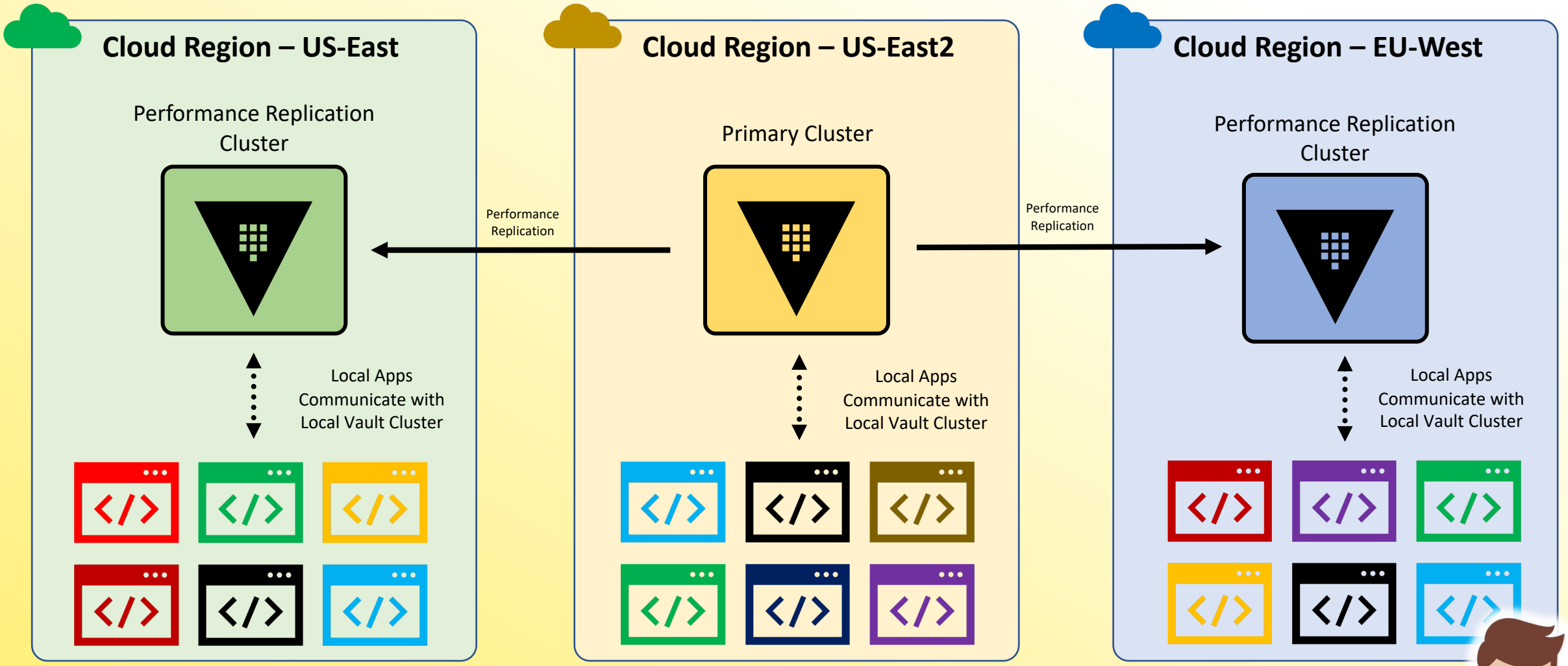
Compare Performance & DR Replication



Replication Architecture



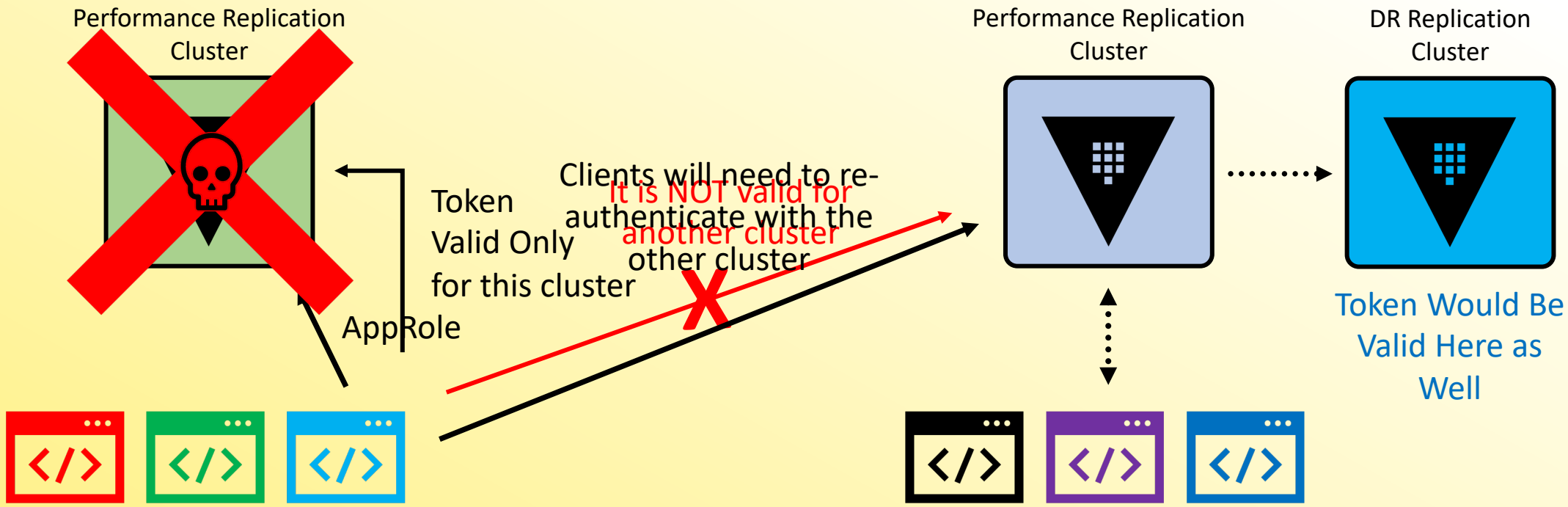
Application Communication



Performance Replication

- Provides active/active solution for applications running in multiple data centers
- Applications authenticate with the LOCAL Vault cluster. Tokens are created and maintained on each cluster and are not replicated via Perf Replication
- If a cluster becomes unavailable and you failover, applications will need to reauthenticate with the new Vault cluster
    - Exception here is if you failover to a DR cluster

Performance Replication



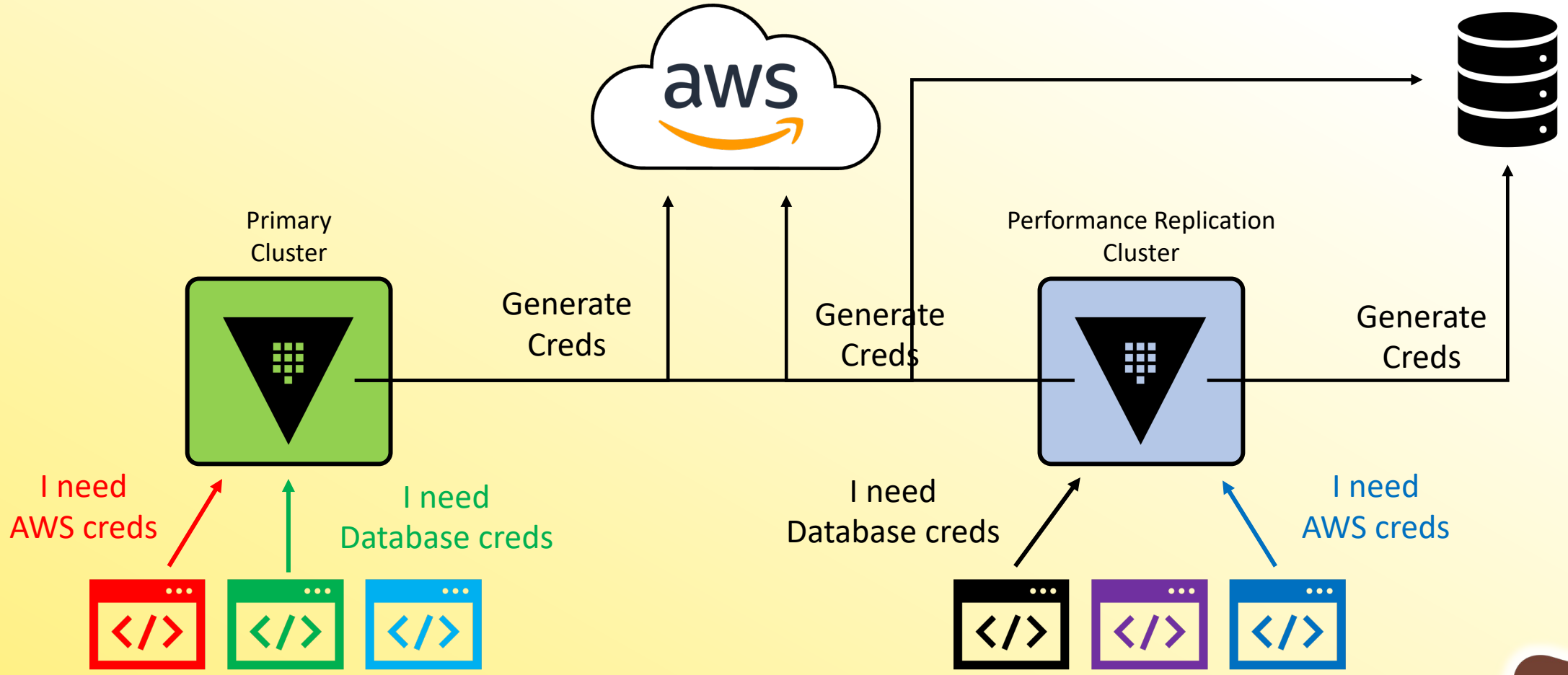
Performance Replication

- Performance replicated clusters handle the retrieval of secrets and the generation of dynamic credentials for local clients

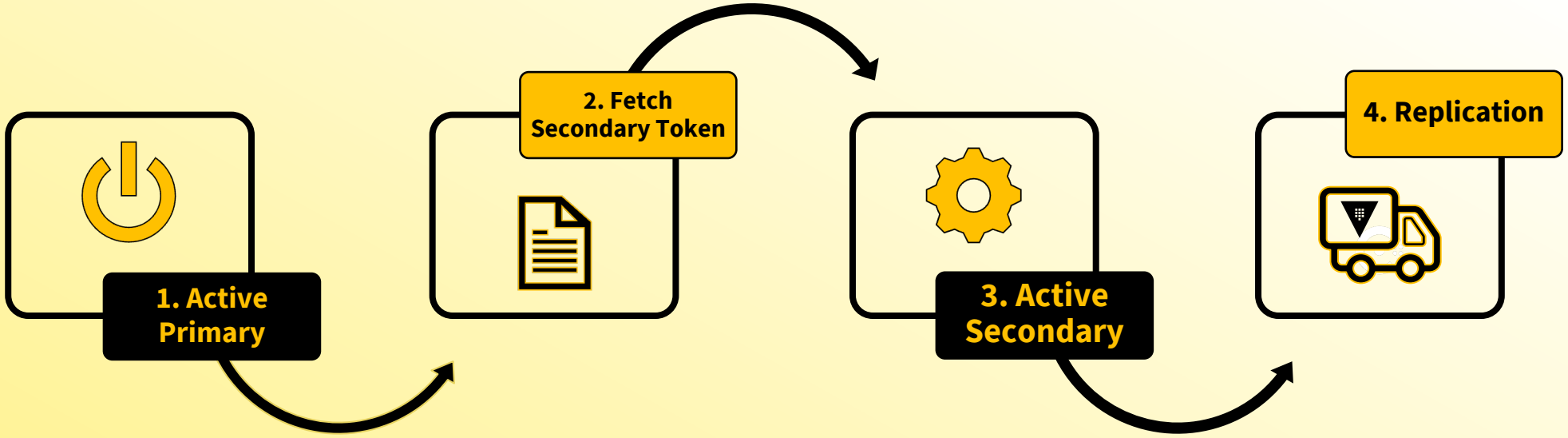
- These requests are handled locally and tokens/leases not replicated to the primary cluster
  - This helps offload some WRITE operations from the primary because the local cluster handles and doesn't forward to the primary cluster

- Any request to write data to the KV, write/updates Vault policies, make Vault configuration changes, etc. WILL be forwarded to the primary

Interaction with External Services



How Do We Set All of this Up?



Activate Performance Replication on the Primary as a Performance Primary

Create a secondary token on the Primary cluster

Activate Performance Replication on the Secondary cluster as a Performance secondary

Watch Vault replicated the data from the Primary to the new Secondary cluster

Configure Replication

1 Activate Performance Replication

```
primary$ vault write -f sys/replication/performance/primary/enable
```

2 Create the Secondary Token

Name it what you want

```
primary$ vault write sys/replication/performance/primary/secondary-token id=<id>
```

3 Activate the Secondary Cluster

Provide token from primary cluster (command above)

```
secondary$ vault write sys/replication/performance/secondary/enable token=<token>
```

Monitor Replication

Check Status of ALL Replication

```
$ vault read -format=json sys/replication/status
```

Check Status of Performance Replication

```
$ vault read -format=json sys/replication/performance/status
```

Performance Replication Only

Check Status of DR Replication

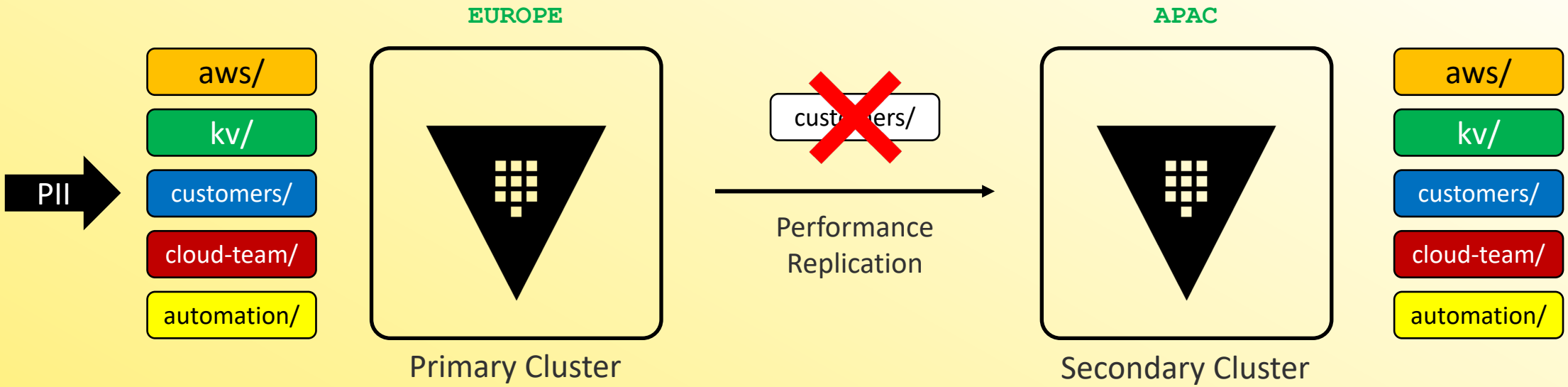
```
$ vault read -format=json sys/replication/dr/status
```

DR Replication Only

Creating a Paths Filter

Replicating Data

Regulatory Compliances may restrict you from replicating certain data (GDPR)



Paths Filter

Vault has a **Paths Filters** capability when using Performance Replication

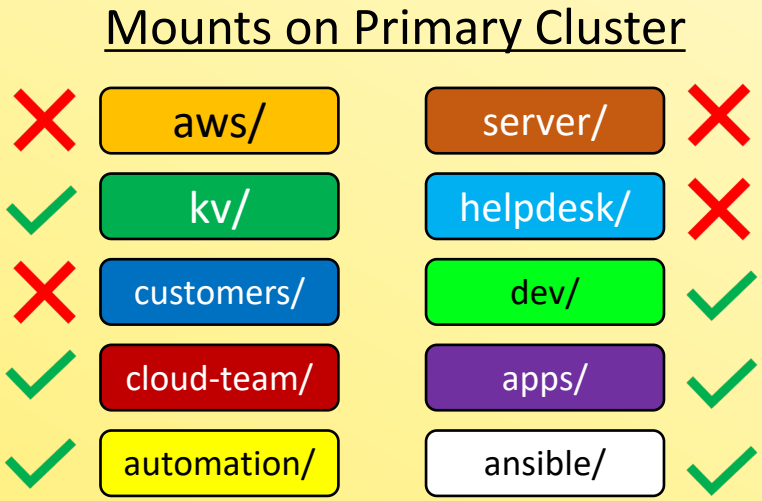
- This enables you to configure an **allowlist** or **denylist** for paths in Vault
- Determines what is replicated to other clusters

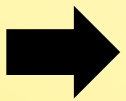
Paths Filters work on paths such as secrets engines, auth methods, AND namespaces

Paths Filter - Allowlist

Allowlist:

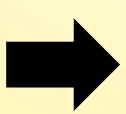
- only the selected paths are included for replication to the secondary



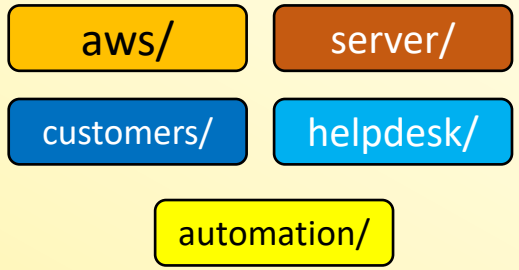


Allowlist

- kv/
- cloud-team/
- dev/
- apps/
- ansible/



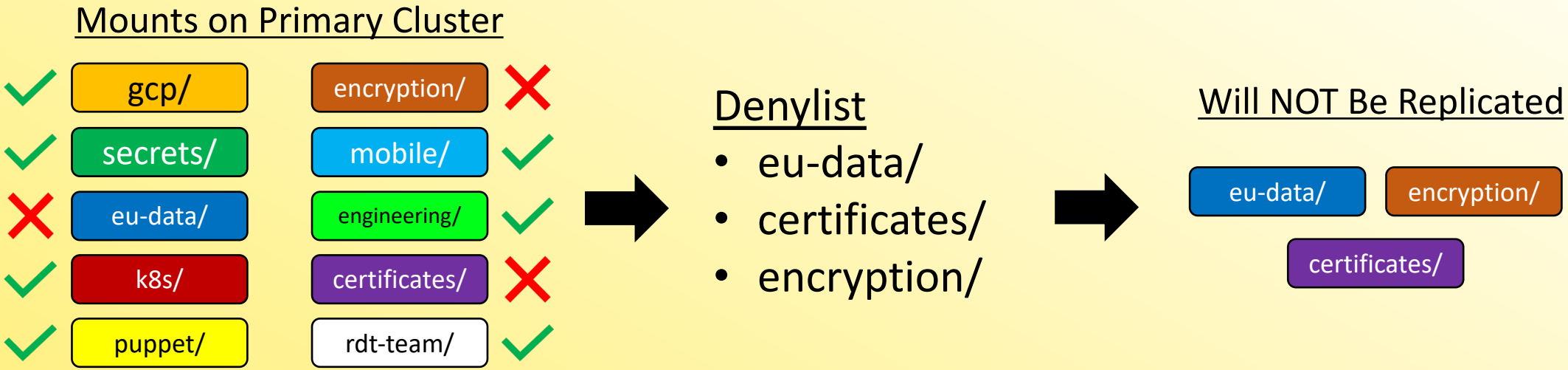
Will Not Be Replicated



Paths Filter - Denylist

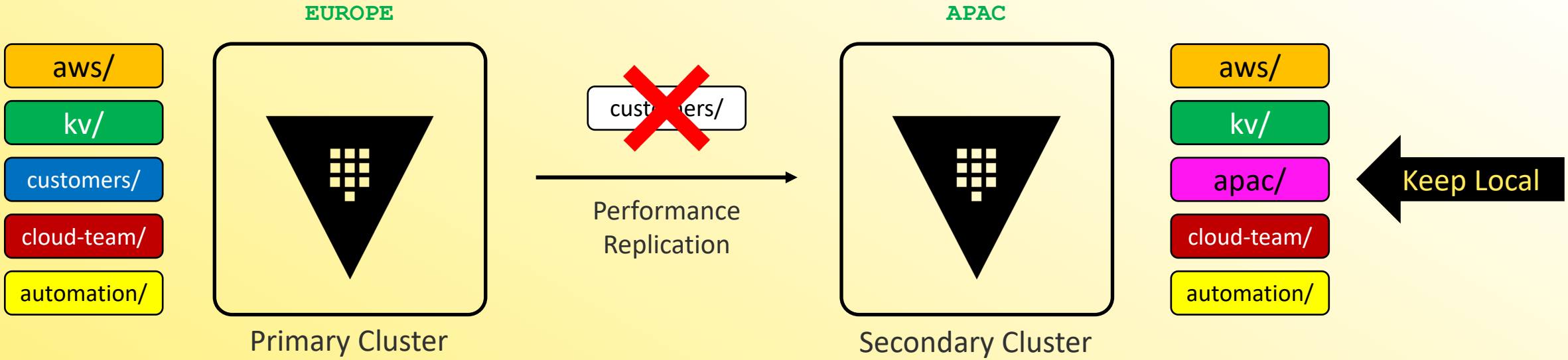
Denylist:

- All paths will be replicated EXCEPT the selected mount paths



Local Mounts

What if we want a mount on the Perf Secondary but we do NOT want it be replicated throughout the replica set?



Create a Local Mount

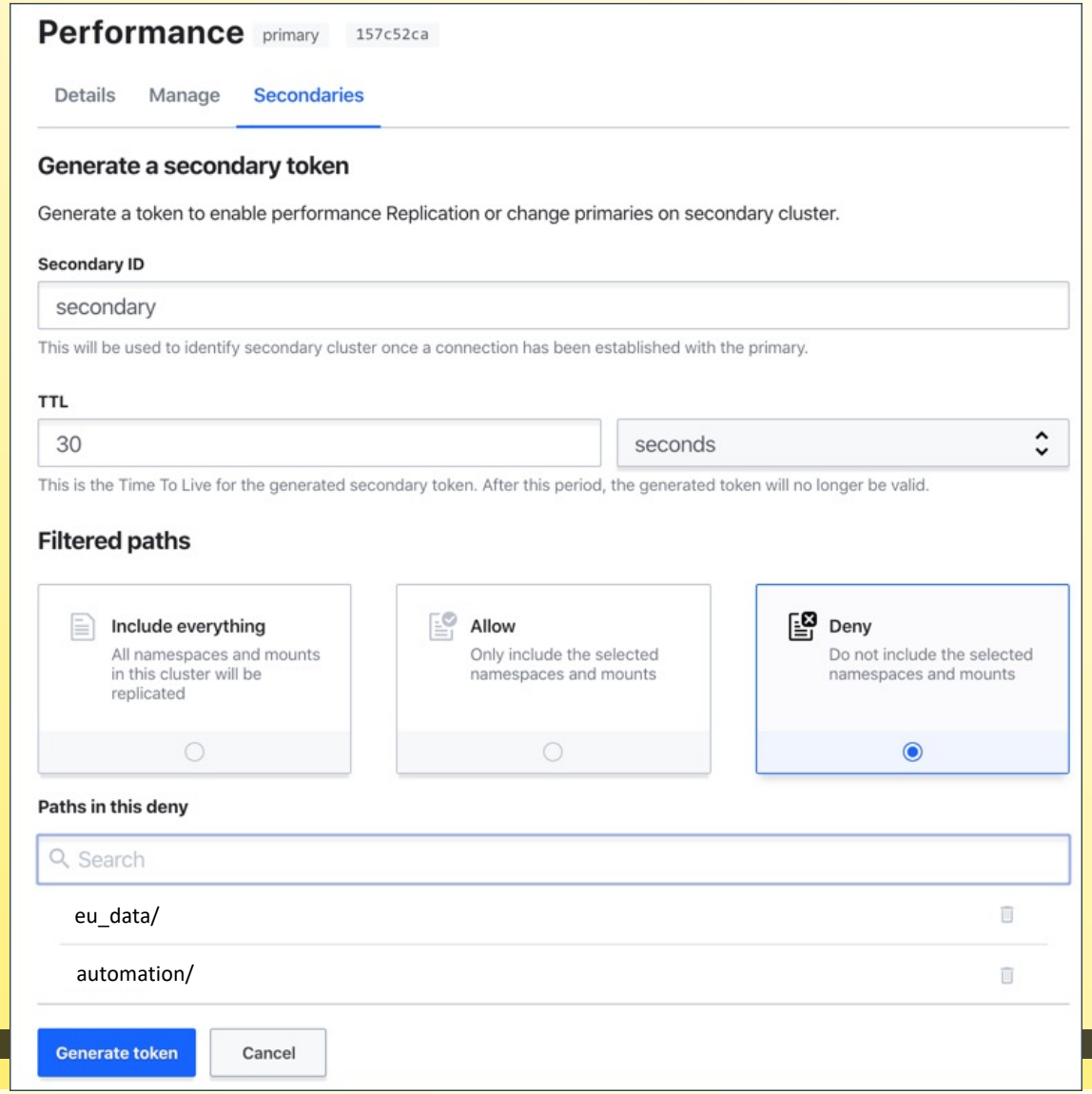
You can mark a secrets engine or auth method as **`local`** so it is not replicated or removed by replication configurations

Enable the secrets engine/auth method on the secondary cluster using the **`-local`** flag

Terminal

```
$ vault secrets enable -local -path=apac kv-v2
```

Create a Paths Filter



Configure within the Secondary configuration section on the Primary Cluster when creating a Secondary token

Select the type of list you want to create – Allow or Deny

Type the path to add

Paths in list

Create a Paths Filter

Terminal

```
$ vault write sys/replication/performance/primary/paths-filter/us-east-dr \
  mode=allow \
  paths=aws/,hcvop/,customers/
```

Allow or Deny List?

Paths to include in the list

The Secondary Cluster ID

Create a Paths Filter

Result of Previous Command