Introduction to vault

- Vault comes with various pluggable components called *secrets engines* and *authentication methods* allowing you to integrate with external systems

- *Vault is available as source code, as a pre-compiled binary, or in packaged formats.*

- *Vault encrypts these secrets prior to writing them to persistent storage,*

Facts About the Dev Mode

- Unseal Key and Root Token values are generated automatically in dev mode
- *Vault automatically unsealed in dev mode however that mode is not recommended for production use case due to its none-persistent use case*
- *Developer mode can be started using* vault server –dev
- *The dev server stores all its data in-memory (but still encrypted), listens on localhost*
- export VAULT_ADDR='http://127.0.0.1:8200’
- export VAULT_TOKEN="s.XmpNPoi9sRhYtdKHaQhkHP6x“
- If the VAULT_TOKEN environment variable set then it will not required vault login command to authenticate vault
- Dev servers have version 2 of KV secrets engine mounted by default

Vault enables Key/Value version2 secrets engine ( ) at the path

Facts About key value store

- Vault enables Key/Value version2 secrets engine (kv-v2) at the path secret/
- Vault retrieve the latest version of the key value when you initiated the command

```
[root@Vault-server-01 ~]# vault kv put secret/hell foo=world
Key              Value
---              -----
created_time     2021-04-04T03:14:57.673256604Z
deletion_time    n/a
destroyed        false
version          1
[root@Vault-server-01 ~]# vault kv put secret/hell foo=world2
Key              Value
---              -----
created_time     2021-04-04T03:15:04.523657563Z
deletion_time    n/a
destroyed        false
version          2
[root@Vault-server-01 ~]# vault kv get secret/hell
====== Metadata ======
Key              Value
---              -----
created_time     2021-04-04T03:15:04.523657563Z
deletion_time    n/a
destroyed        false
version          2

=== Data ===
Key    Value
---    -----
foo    world2
```

Vault enables key/value version2 secrets engine ( ) at the path

Facts About key value store

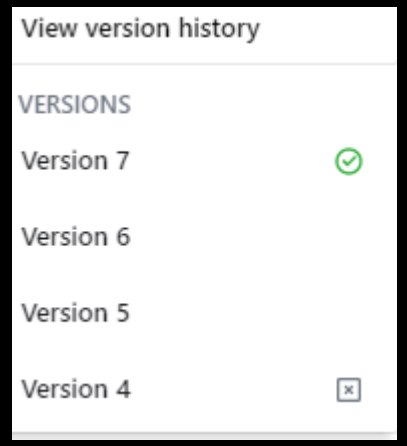
- vault secrets enable -path=secret kv-v2
- Upgrade the kv version 1 to 2 using vault kv enable-versioning secret/

vault kv get -version=5 secret/customer/acme

Retrieve the specific version of the key

vault write secret/config max_versions=4

If you have the seven version created then version 4 to 7 display in the UI



Vault enables Key/Value version2 secrets engine ( ) at the path

Facts About key value store

- Delete the KV not delete it associate metadata

```
[root@Vault-server-01 ~]# vault kv delete secret/hell
Success! Data deleted (if it existed) at: secret/hell
```

- vault kv metadata get secret/hell

```
[root@Vault-server-01 ~]# vault kv  metadata  get secret/hell
========== Metadata ==========
Key                     Value
---                     -----
cas_required            false
created_time            2021-04-04T03:14:57.673256604Z
current_version         2
delete_version_after    0s
max_versions            0
oldest_version          0
updated_time            2021-04-04T03:15:04.523657563Z

====== Version 1 ======
Key              Value
---              -----
created_time     2021-04-04T03:14:57.673256604Z
deletion_time    n/a
destroyed        false

====== Version 2 ======
Key              Value
---              -----
created_time     2021-04-04T03:15:04.523657563Z
deletion_time    2021-04-04T03:20:11.881453291Z
destroyed        false
```

Vault enables Key/Value version2 secrets engine ( ) at the path

Facts About key value store

- Custom Path can be enabled

```
[root@Vault-server-01 ~]# vault secrets enable -path=kv kv
Success! Enabled the kv secrets engine at: kv/
[root@Vault-server-01 ~]# vault secrets enable -path=mykv kv
Success! Enabled the kv secrets engine at: mykv/
[root@Vault-server-01 ~]#
```

- Vault secret list the path

```
[root@Vault-server-01 ~]# vault secrets list
Path          Type         Accessor              Description
----          ----         --------              -----------
cubbyhole/    cubbyhole    cubbyhole_b23e28d9    per-token private secret storage
identity/     identity     identity_e833556d     identity store
kv/           kv           kv_2d8dc89e           n/a
mykv/         kv           kv_4560c137           n/a
secret/       kv           kv_cff1a41b           key/value secret storage
sys/          system       system_e6ae5273       system endpoints used for control, policy and debugging
```

Vault enables Key/Value version2 secrets engine ( ) at the path

Facts About secret engine

- When a secrets engine is disabled, all secrets are revoked and the corresponding Vault data and configuration is removed

- Disable secret engine will disable only specific mounted secret engine it wont effect other multiple version of same secret engine type

Vault enables key/value version2 secrets engine ( ) at the path

Facts About path

- There to-way to enable the secret engine vault secrets enable -path=kv kv
  vault secrets enable kv

- Other path can be configured on the same secret engine using
  Vault secret enable –path=mykv kv

- The sys/ path corresponds to the system backend.

- Path-help can be available after the secret engine activated

- same type of auth method at different paths

- The path prefix tells Vault which secrets engine to which it should route traffic.

- Each path is completely isolated and cannot talk to other paths.

Vault enables Key/Value version2 secrets engine ( ) at the path

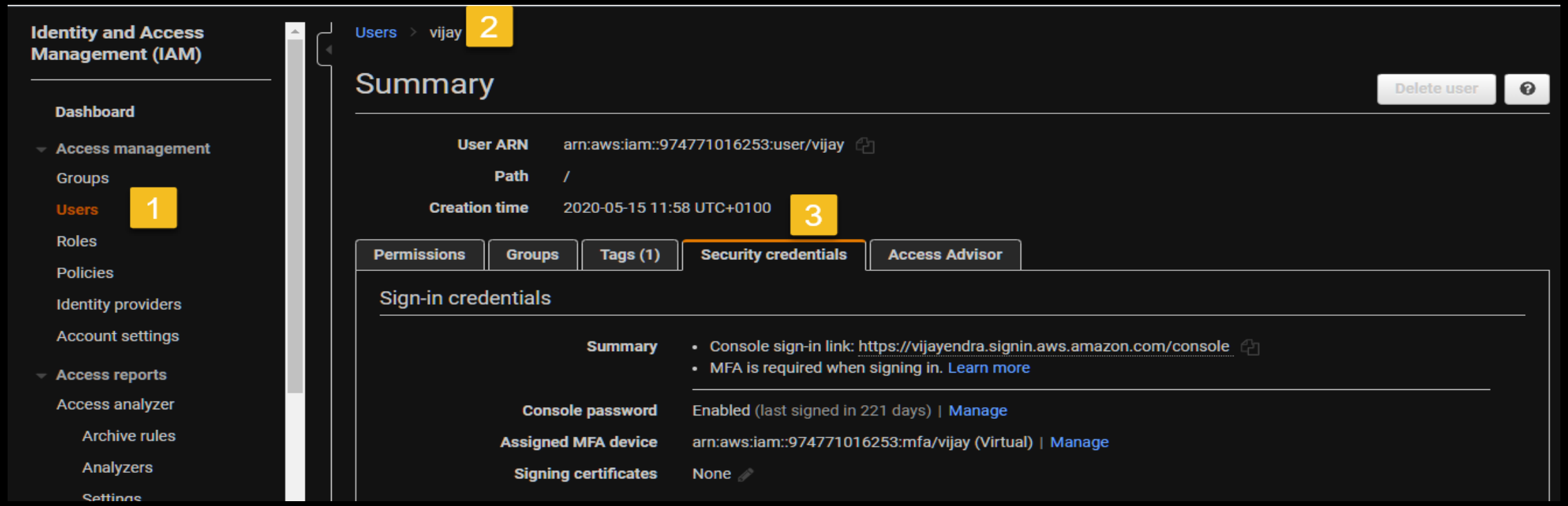
Facts About dynamic secrets

- dynamic secrets are generated when they are accessed
- dynamic secrets can be revoked immediately after use

Vault enables Key/Value version2 secrets engine ( ) at the path

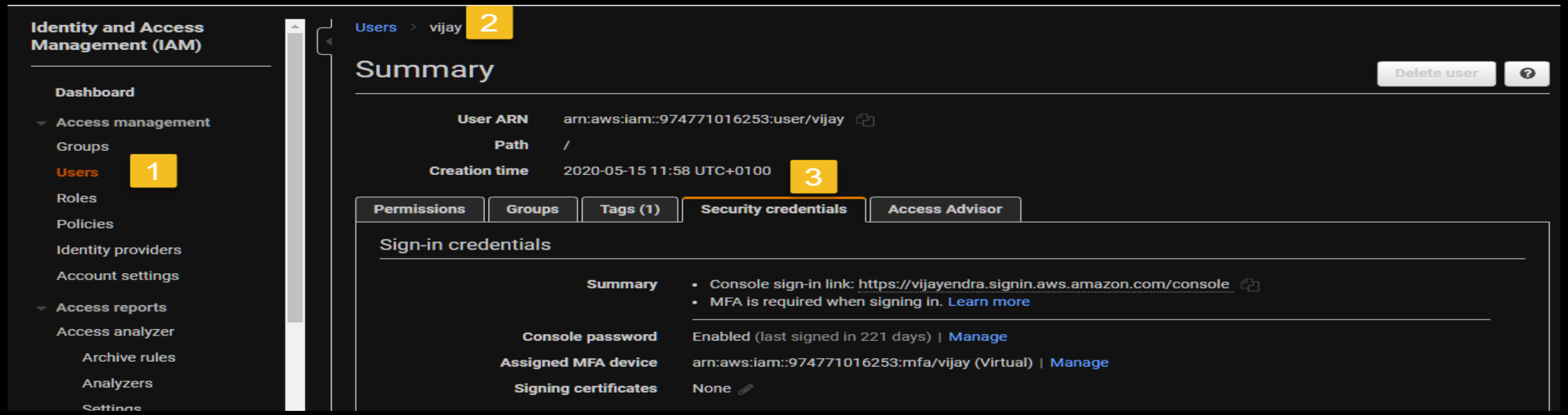
AWS Secret Engine

Step-1 Create the IAM user with administrator access in the AWS

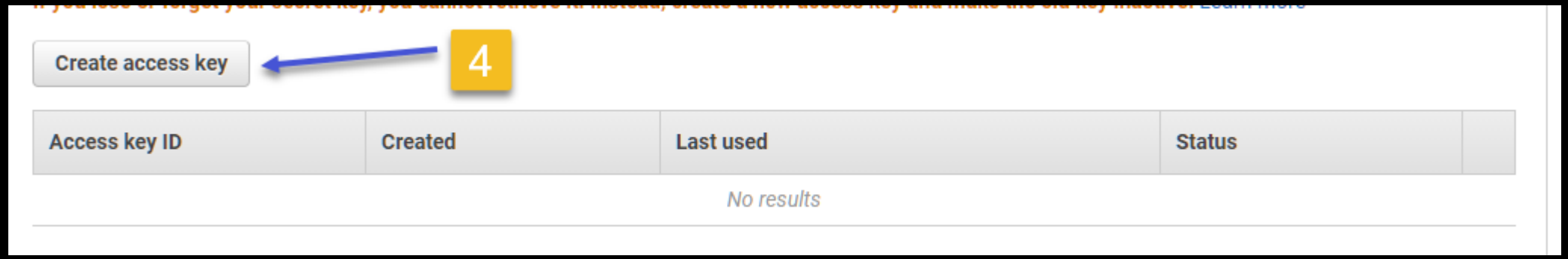


AWS Secret Engine

Step-2 click on the user which hold the admin rights in the console and tap on security credential



Step-3 create the access key



Vault enables key/value version2 secrets engine ( ) at the path

AWS Secret Engine

Step-4 set the access key and key id

```
root@Vault-server-01 ~]# export AWS_ACCESS_KEY_ID=AKIA6F5HBKI6WKZ22O5J
root@Vault-server-01 ~]# export AWS_SECRET_ACCESS_KEY=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
```

Step-5 configure the aws secret engine

```
[root@Vault-server-01 ~]# vault write aws/config/root  \
> access_key=$AWS_ACCESS_KEY_ID  \
> secret_key=$AWS_SECRET_ACCESS_KEY
Success! Data written to: aws/config/root
[root@Vault-server-01 ~]#
```

Step-6 create the role for the IAM user but not in aws

```
[root@Vault-server-01 ~]# vault write aws/roles/my-role \
>         credential_type=iam_user \
>         policy_document=-<<EOF
> {
>   "Version": "2012-10-17",
>   "Statement": [
>     {
>       "Sid": "Stmt1426528957000",
>       "Effect": "Allow",
>       "Action": [
>         "ec2:*"
>       ],
>       "Resource": [
>         "*"
>       ]
>     }
>   ]
> }
> EOF
```

this role allows all EC2 Action

AWS Secret Engine

Step-7 Secret key and Access key is automatically created using vault

```
[root@Vault-server-01 ~]# vault read aws/creds/my-role
Key                Value
---                -----
lease_id           aws/creds/my-role/3mVtOFiYqv6cXgXQ0711je9R
lease_duration     768h
lease_renewable    true
access_key         AKIA6F5HBKI626WNGQN3
secret_key         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
security_token     <nil>
[root@Vault-server-01 ~]#
```

Step-8 vault will create random user with associate with access key and mentioned IAM Policy

Find users by username or access key

User name	Groups	Access key age	Last activity	Creation time	Group count
raj	None	None	221 days	2020-08-25 12:33 UTC+0100	0
rohit	None	None	247 days	2020-05-08 11:49 UTC+0100	0
vault-root-m...	None	Today	None	2021-04-04 05:37 UTC+0100	0
vijay	None	Today	221 days	2020-05-15 11:58 UTC+0100	0

Vault enables Key/Value version2 secrets engine ( ) at the path

AWS Secret Engine

Step-9 once the lease is revoked the credential get deleted from aws

```
[root@Vault-server-01 ~]#
[root@Vault-server-01 ~]# vault lease revoke aws/creds/my-role/3mVt0FiYqv6cXgXQ0711je9R
All revocation operations queued successfully!
[root@Vault-server-01 ~]#
```

Step-10 user deleted from aws console

User name	Groups	Access key age	Password age	Last activity	MFA	Crea
raj	None	None	221 days	221 days	Not enabled	202
rohit	None	None	247 days	247 days	Not enabled	202
vijay	None	None	323 days	221 days	Virtual	202

Github Integration

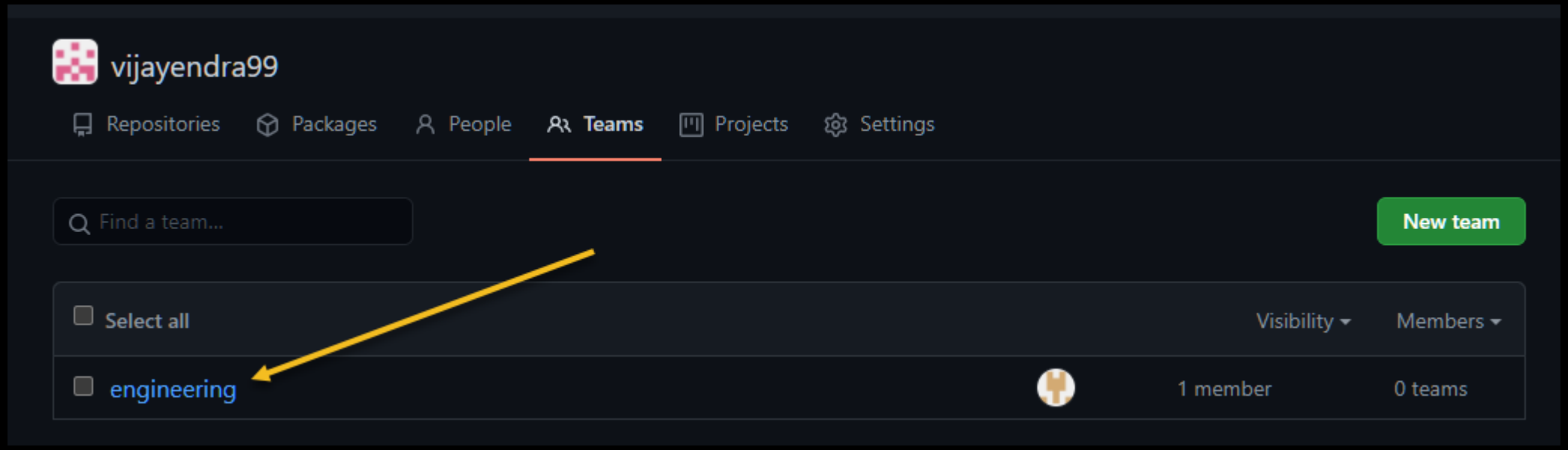
Step-1 vault auth enable github

Step-2 vault write auth/github/config orgnization=vijayendra99

Organization configured under the GitHub account

Github Integration

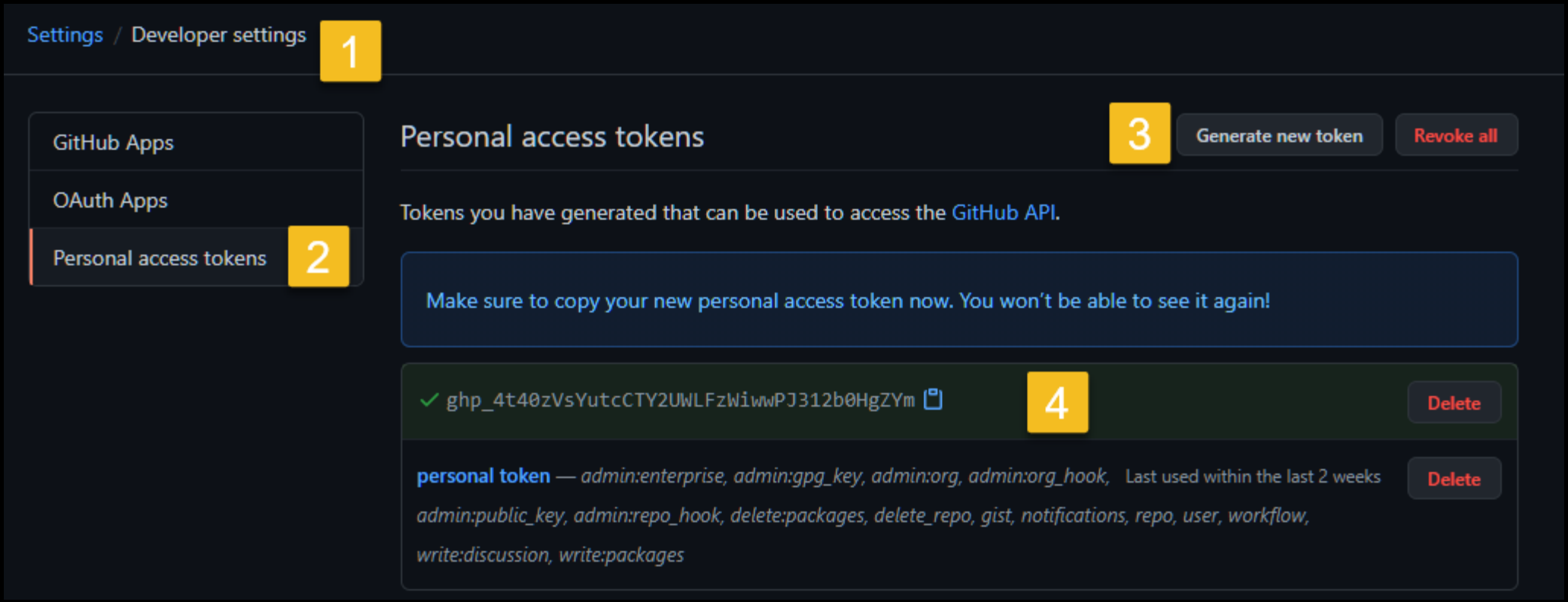
Step-3 vault write auth/github/map/teams/engineering value=default.applications



Team Created under the organization

Github Integration

Step-3 generate the person access token on the github



Vault enables Key/Value version2 secrets engine ( ) at the path

Github Integration

Step-4 vault login --method=github

```
[root@Vault-server-01 ~]# vault login -method=github
GitHub Personal Access Token (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.UFiINxWX7harqMRnWpC7piAz
token_accessor         gwfpAUrGWiq08R1sCpm0tqaC
token_duration         768h
token_renewable        true
token_policies         ["applications" "default"]
identity_policies      []
policies               ["applications" "default"]
token_meta_username    vijayendra99099
token_meta_org         vijayendra99
```

Authenticated with github personal access token

Vault enables Key/Value version2 secrets engine ( ) at the path

Facts About policies

- Policies are authored in HCL, but are JSON compatible.
- the root and default policies are required policies and cannot be deleted
- Restrict the use of root policy, and write fine-grained policies to practice least privileged.
- Two way to write the Policy using command line
- vault policy write my-policy /tmp/policy.hcl
- cat my-policy.hcl | vault policy write my-policy -

Vault enables Key/Value version2 secrets engine ( ) at the path

Facts About policies

vault read policy default [check the capabilities according to its structure]

Create the token with associate policies

vault token create -policy=my-policy

```
[root@Vault-server-01 ~]# vault token create -policy=my-policy
Key                  Value
---                  -----
token                s.afxp8fyfAsx3z4gYhqBEApqc
token_accessor       sc8gRfM0NjFMFnOe27gsjyb9
token_duration       768h
token_renewable      true
token_policies       ["default" "my-policy"]
identity_policies    []
policies             ["default" "my-policy"]
[root@Vault-server-01 ~]#
```

Vault enables Key/Value version2 secrets engine ( ) at the path

Facts About policies

```
path "secret/data/*" {
  capabilities = ["create", "update"]
}
```

```
[root@Vault-server-01 ~]# vault kv put secret/creds/mykv/kv1 password="mypassword1"
Key                Value
---                -----
created_time       2021-04-04T06:11:50.065797494Z
deletion_time      n/a
destroyed          false
version            1
[root@Vault-server-01 ~]# vault kv put secret/creds/mykv/kv2/test password="mypassword1"
Key                Value
---                -----
created_time       2021-04-04T06:15:40.094050509Z
deletion_time      n/a
destroyed          false
version            1
[root@Vault-server-01 ~]# vault kv put secret/creds/mykvnew/kv2/test password="mypassword1"
Key                Value
---                -----
created_time       2021-04-04T06:15:54.892577616Z
deletion_time      n/a
```

many path can be created under secret/data/ as wildcard permission given in policy to create and update it

Vault enables Key/Value version2 secrets engine ( ) at the path

Facts About policies

```
path "secret/data/foo" {
  capabilities = ["read"]
}
```

```
[root@Vault-server-01 ~]# vault kv put secret/foo  a=b
Error writing data to secret/data/foo: Error making API request.

URL: PUT http://127.0.0.1:8200/v1/secret/data/foo
Code: 403. Errors:

* 1 error occurred:
        * permission denied
```

Only read permission for the specific path due explicitly defined specific path to read the content only

Vault enables key/value version2 secrets engine ( ) at the path

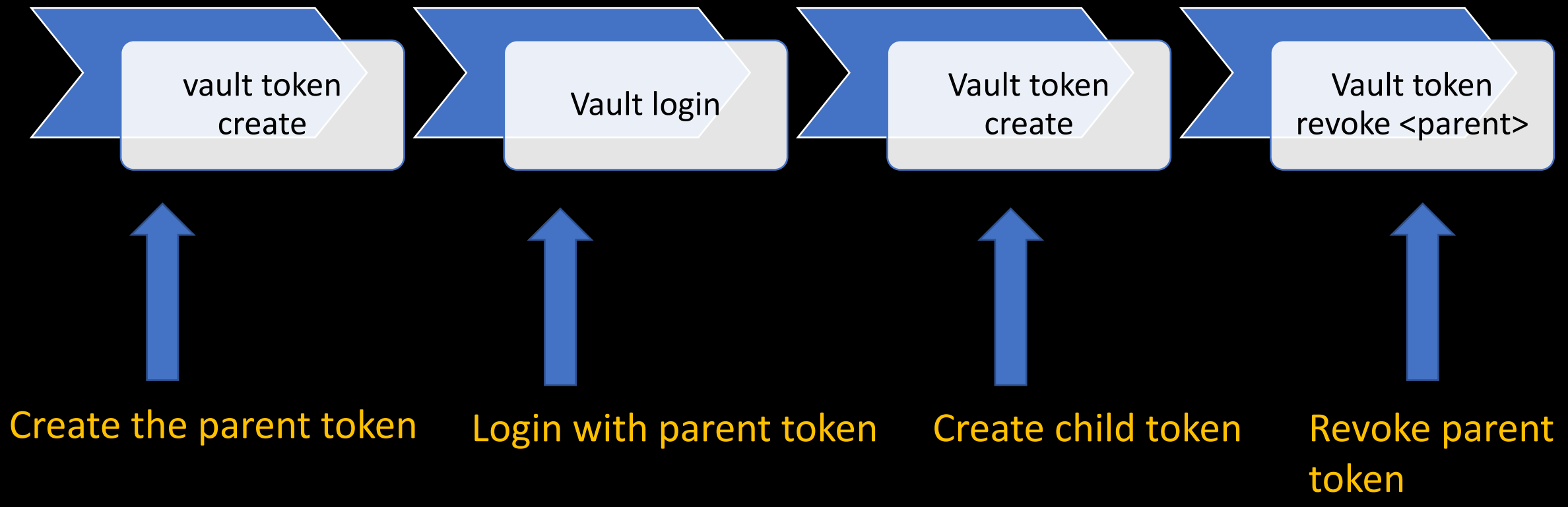
Facts About key value

- KV v2 secrets engine using the vault kv CLI commands, you can omit /data in the secret path.

- the root and default policies are required policies and cannot be deleted

Facts About token

- Child token inheritance the policies from its parents
- revoke parent also revoke child token if it is created using the parent token



Vault enables key/value version2 secrets engine ( ) at the path

Facts About token

- Tokens are the core method for authentication within Vault
- There are two types of Vault tokens: service token and batch token
- Tokens can be used directly or dynamically generated by the auth methods.

Feature	Service Token	Batch token
Persistent in backend	YES	NO
Renew and Revoke the token	YES	NO
Flexibility and feature	YES	NO
Lightweight and scalable	NO	YES
Can be Root Token	YES	NO
Can be perodic	YES	NO
Use in Performance Replication	NO	YES
Has A cubbyhole	YES	NO

Facts About token

- Service token lifecycle
- s.b519c6aa... (1h)
- |___ s.6a2cf3e7... (4h)
- |___ s.1d3fd4b2... (2h)
- |___ s.794b6f2f... (3h)
- When the token with 2 hour is expired it will automatically discard token underneath token no-matter it still have TTL

- TTL vs Max TTL

Token can be renewed in every 30 minute up to 3 Hour of MAX TTL and once the MAX TTL reached it cant be renewed

30 Min TTL	30 Min TTL	30 Min TTL	30 Min TTL	30 Min TTL	30 Min TTL

Note : MAX TTL can be anything from 1 hour to 24 Hour whichever decided by admin

Facts About token

- Maximum TTL is 32 days by default however you can increase it using vault server configuration file

- VAULT_TOKEN environment variable set, the CLI commands will always use this value
- Tokens with use limit
- vault token create -ttl=1h -use-limit=2 -policy=default

Token can be used 2 time within 1 hour time limit

Token can be used up to 2 times

Facts About token

- **Periodic service tokens**
- Root or sudo users have the ability to generate periodic tokens
- Periodic tokens have a TTL (validity period), but no max TTL
- useful for long-running services that cannot handle regenerating a token.
- You can renew the generated token indefinitely for as long as it does not expire
- vault token lookup s.6XhDGuPwiJgCbQUIIei4uA1Z

Find details about the token type

Facts About token

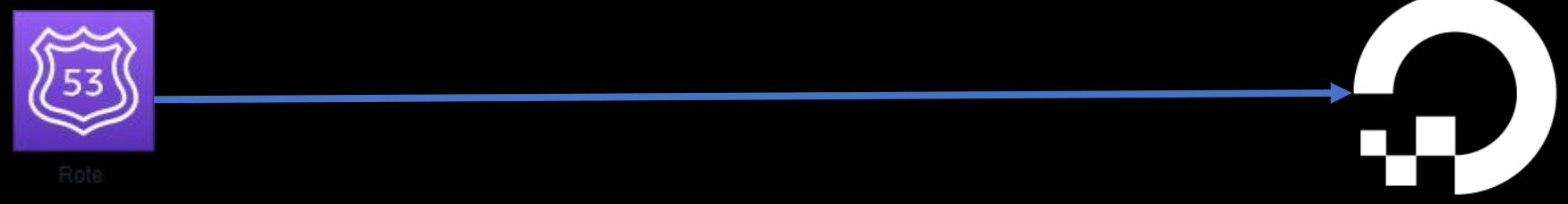
- **Orphan tokens**
- **Orphan tokens are not children of their parent; therefore, orphan tokens do not expire when their parent does**
- **Orphan tokens still expire when their own max TTL is reached.**

Facts About token

- **Orphan tokens**
- **Orphan tokens are not children of their parent; therefore, orphan tokens do not expire when their parent does**
- **Orphan tokens still expire when their own max TTL is reached.**

Create self signed certificate using AWS Route 53 and Map to Digital Ocean

A Record vault-01.examsimulator.net = 159.65.147.103



Route53

Digital ocean

159.65.147.103

step -1 yum -y install https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm

Step-2 yum install certbot

Step-3 certbot certonly --standalone -d vault-01.examsimulator.net

Step-4 cp /etc/letsencrypt/live/vault-01.examsimulator.net/fullchain.pem /etc/vault.d/vault_cert.crt
cp /etc/letsencrypt/live/vault-01.examsimulator.net/privkey.pem /etc/vault.d/vault_cert.key

Vault Deployment using Raft and Self Signed Certificate

```
storage "raft" {
  path    = "/etc/vault.d/data"
  node_id = "vault-server-01"
}

listener "tcp" {
  address     = "0.0.0.0:8200"
  tls_cert_file = "/etc/vault.d/example.crt"
  tls_key_file  = "/opt/vault.d/example.key"
  tls_disable = 0
}

api_addr = "http://127.0.0.1:8200"
cluster_addr = "https://127.0.0.1:8201"
ui = true
```

Directory need to create whare raft data will be stored

Use self signed certificated
which is generated during vault installation and enable TLS for the vault

Note: it will open web url on https: but certificate can be validate as it is local

Vault enables key/value version2 secrets engine ( ) at the path

Facts About seal/unseal/key

- Vault does not store any of the unseal key shards
- Shamir's Secret Sharing to split the master key into shards
- Vault uses the encryption key to encrypt data at rest in a storage backend like the filesystem or Consul
- The process for generating a new master key and applying Shamir's algorithm is called "rekeying
- The process for generating a new encryption key for Vault to encrypt data at rest is called "rotating".
- Also notice that the unseal process is stateful. You can go to another computer, use vault operator unseal, and as long as it's pointing to the same server, that other computer can continue the unseal process these also applicable to UI version
- As a root user, you can reseal the Vault with vault operator seal

Vault enables Key/Value version2 secrets engine ( ) at the path

Re-keying

vault operator rekey -init -key-shares=3 -key-threshold=2

```
Key                     Value
---                     -----
Nonce                   0844c65c-e37e-c228-0a70-e59d678810b1
Started                 true
Rekey Progress          0/2
New Shares              3
New Threshold           2
Verification Required   false
```

vault operator rekey -nonce=0844c65c-e37e-c228-0a70-e59d678810b1

```
[root@Vault-server-01 ~]# vault operator rekey -nonce=0844c65c-e37e-c228-0a70-e59d678810b1
Rekey operation nonce: 0844c65c-e37e-c228-0a70-e59d678810b1
Unseal Key (will be hidden):
Key                     Value
---                     -----
Nonce                   0844c65c-e37e-c228-0a70-e59d678810b1
Started                 true
Rekey Progress          1/2
New Shares              3
New Threshold           2
Verification Required   false
```

Supply your existing unseal key to get the new unseal key

Vault enables Key/Value version2 secrets engine ( ) at the path

Re-keying

vault operator rekey -nonce=0844c65c-e37e-c228-0a70-e59d678810b1

```
[root@Vault-server-01 ~]# vault operator rekey -nonce=0844c65c-e37e-c228-0a70-e59d678810b1
Rekey operation nonce: 0844c65c-e37e-c228-0a70-e59d678810b1
Unseal Key (will be hidden):

Key 1: uVXW5Xxyy3plNyx3iqA1Hw6N6nFzAQ3S1yAgnNcPcvqc
Key 2: 2C/spKFqxYRb9rij/JcathQXdh/oHW720SMTrxbf3YqR
Key 3: 42F09IyOoFPtEZlUwPuSsOdTC8cq10Kz6LJR7fEG4o9l
```

In the second time it will generate the 3 unique unseal key using

Note Re-keying will generate the new unseal key

Vault enables Key/Value version2 secrets engine ( ) at the path

Rotating encryption key

vault operator generate-root -init

```
[root@Vault-server-01 ~]# vault operator generate-root -init
A One-Time-Password has been generated for you and is shown in the OTP field.
You will need this value to decode the resulting root token, so keep it safe.
Nonce         4d534311-4396-1b49-1d8f-d5dd140d88aa
Started       true
Progress      0/2
Complete      false
OTP           PKXKTer5HpG4FhCG08nE9jAVSS
OTP Length    26
```

vault operator generate-root \
-nonce=4d534311-4396-1b49-1d8f-d5dd140d88aa

```
[root@Vault-server-01 ~]# vault operator generate-root \
> -nonce=4d534311-4396-1b49-1d8f-d5dd140d88aa
Operation nonce: 4d534311-4396-1b49-1d8f-d5dd140d88aa
Unseal Key (will be hidden):
Nonce        4d534311-4396-1b49-1d8f-d5dd140d88aa
Started      true
Progress     1/2
Complete     false
```

Rotating encryption key

vault operator generate-root \
-nonce=4d534311-4396-1b49-1d8f-d5dd140d88aa

```
[root@Vault-server-01 ~]# vault operator generate-root -nonce=4d534311-4396-1b49-1d8f-d5dd140d88aa
Operation nonce: 4d534311-4396-1b49-1d8f-d5dd140d88aa
Unseal Key (will be hidden):
Nonce            4d534311-4396-1b49-1d8f-d5dd140d88aa
Started          true
Progress         2/2
Complete         true
Encoded Token    I2UtLW1TAUEKRgsCAAp3FUoLCzMPGy8hJ2M
```

vault operator generate-root -decode=I2UtLW1TAUEKRgsCAAp3FUoLCzMPGy8hJ2M \
-otp=PKXKTer5HpG4FhCG08nE9jAVSS

s.uf96stB6L6Fb4Rz3ev6qnwt0

Note rotation will generate the root token

Vault enables key/value version2 secrets engine ( ) at the path

Seal/Unseal/Auto-unseal

The initialization generates recovery keys (instead of unseal keys) when using auto-unseal

The seal can be migrated from Shamir Seal to Auto Unseal

The seal can be migrated from Shamir Seal to Auto Unseal,
Auto Unseal to Shamir Seal, and Auto Unseal to another Auto Unseal

Hashicorp vault replication

Hashicorp vault replication

- providing consistency, scalability, and highly-available disaster recovery

.

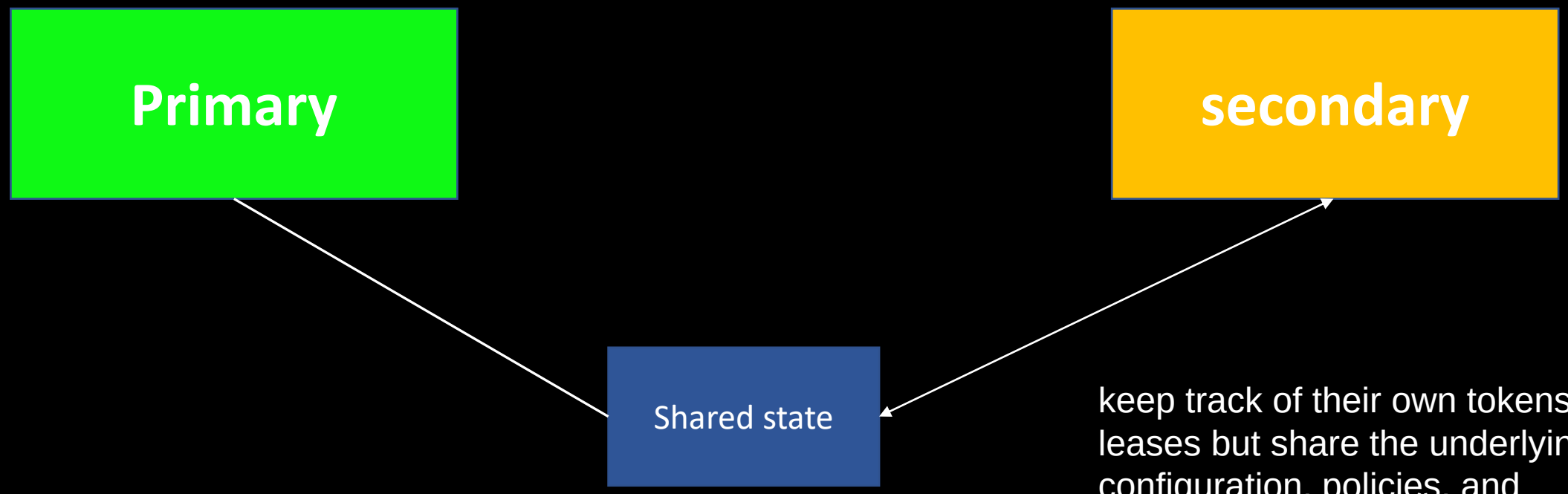
- Multiple Vault clusters communicate in a one-to-many near real-time flow
- Leader known as primary and follower known as secondary
- All communication are end to end encrypted using the MTLS
- Performance replication and disaster recovery are the type of the replication
- When a cluster is marked as the primary it generates a self-signed CA certificate
- once the token is used successfully (in this case, to activate a secondary) it is useless after it use
- all Vault information is replicated from the primary to secondaries except for tokens and secret leases in performance replication
- when replication is enabled, all of the secondary's existing storage will be wiped.
- activating as a secondary will be the first thing that is done upon setting up a new cluster for replication

Hashicorp vault replication

- Local mounts do not propagate data from the primary to secondaries
- In DR replication, secondary clusters do not forward service read or write requests until they are promoted and become a new primary
- You need a DR operation token to promote the cluster
- DR operation token must be executed by each unseal key holder after the successful token
- Vault does not support an automatic failover/promotion of a DR secondary cluster

eep track of their own tokens and leases but share the underlying configuration, policies, and supporting secrets (K/V values, encryption keys for

Performance cluster

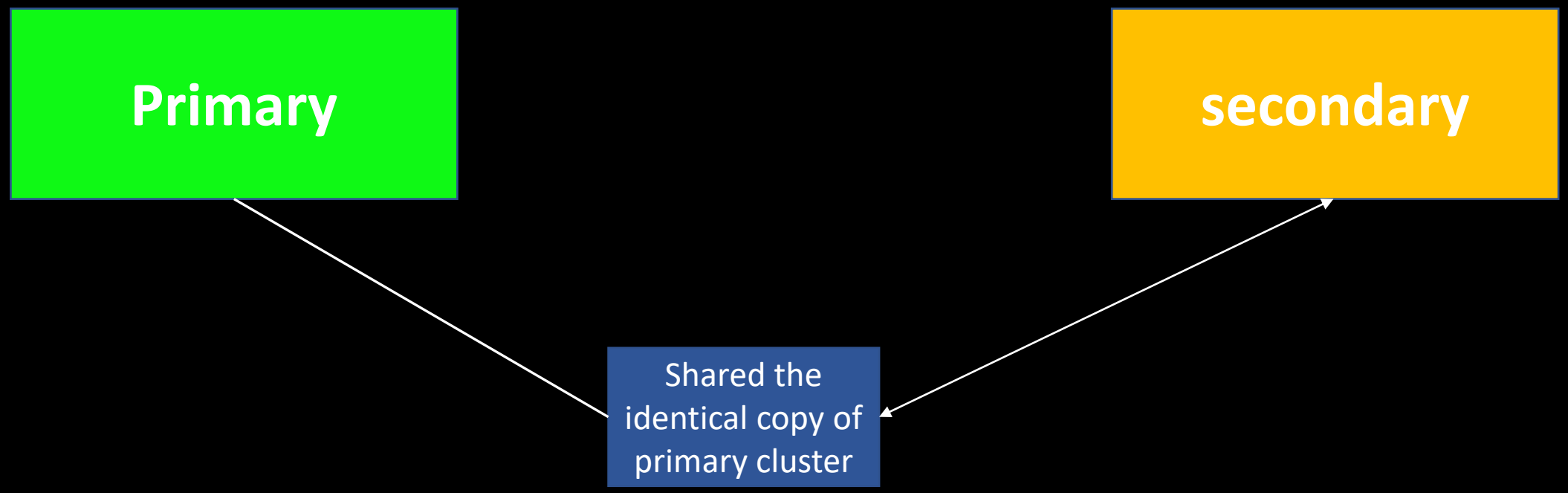


keep track of their own tokens and leases but share the underlying configuration, policies, and supporting secrets K/V values, encryption keys for transit

If the share state modify then secondary send the request to primary

eep track of their own tokens and leases but share the underlying configuration, policies, and supporting secrets (K/V values, encryption keys for

Disaster Recovery cluster



eep track of their own tokens and leases but share the underlying configuration, policies, and supporting secrets (K/V values, encryption keys for

Are my DR Clusters in Sync?

- state of secondary will be stream-wals
- last_remote_wal on the secondary should match (or be very close to) the last_wal on the primary
- Generally, the Merkle root on the primary and secondary will match

Response-Wrapping Tokens

Response-Wrapping Tokens

- When a response is wrapped, the normal API response from Vault does not contain the original secret

TTL
Token
Creation Time
Creation Path
Wrapped Accessor

+ Response-Wrapping Tokens

- response wrapped token is built for single-use token
- It provides cover by ensuring that the value being transmitted across the wire is not the actual secret but a reference to such a secret
- only a single party can ever unwrap the token and see what's inside.
- It limits the lifetime of secret exposure because the response-wrapping token has a lifetime that is separate from the wrapped secret (and often can be much shorter), so if a client fails to come up and unwrap the token, the token can expire very quickly.

Response-Wrapping Token Operations

Lookup	sys/wrapping/lookup	Validate creation time, creation path, and TTL.
unwrap	sys/wrapping/unwrap	Wired format response.
rewrap	sys/wrapping/rewrap	Used for long lived secret.
wrap	sys/wrapping/wrap	echoes back the data sent to it in a response-wrapping.

Response-Wrapping Token Creation

1 The original HTTP response is serialized

2 A new single-use token is generated with the TTL supplied by the client

3 internally, the original serialized response is stored in the single-use token's cubbyhole

4 A new response is generated, with the token ID, TTL, and path stored in the new response's wrap information object

5 The original HTTP response is serialized

Lease, Renew, and Revoke

- Vault always returns a lease_id to renew and revoke the lease for the secret
- Lease can be increment using the below mention command ➔ **vault lease renew -increment=3600 my-lease-id**
- **to revoke all AWS access keys, you can do vault lease revoke -prefix aws/**

Tokens

- there are two types of tokens: service tokens and batch tokens.
- The initial root token generated at vault operator init time -- this token has no expiration
- a root token with an expiration cannot create a root token that never expires

main Token

When token holder create another token then that token called child token and if the main token deleted by the admin then corresponding token also get deleted

Child Token

To overcome these issue orphan token can be used or created

Orphan Tokens

- ❑ Via write access to the auth/token/create-orphan endpoint
- ❑ By having sudo or root access to the auth/token/create and setting the no_parent parameter to true
- ❑ Via token store roles

Token Accessors

- ❑ When tokens are created, a token accessor is also created and returned. This accessor is a value that acts as a reference to a token

1. Look up a token's properties (not including the actual token ID)
2. Look up a token's capabilities on a path
3. Renew the token
4. Revoke the token
5. Audit device and service to service communication

Token Time-To-Live, Periodic Tokens, and Explicit Max TTLs

- Every token excepts root token have TTL period which indicate as validity
- maximum TTL value is dynamically generated and can change from renewal to renewal,
- The system max TTL, which is 32 days but can be changed in Vault's configuration file.
- a long-running service needs to maintain its SQL connection pool over a long period of time. In this scenario, a periodic token can be used.
- At issue time, the TTL of a periodic token will be equal to the configured period. At every renewal time, the TTL will be reset back to this configured period,
- if the system stops renewing within this period the token will expire relatively quickly.
- A token with both a period and an explicit max TTL will act like a periodic token but will be revoked when the explicit max TTL is reached
- Once the parent token expires, so do all its children regardless of their own TTLs.

Token Time-To-Live, Periodic Tokens, and Explicit Max TTLs

- if a token's TTL is 30 minutes and the maximum TTL is 24 hours, you can renew the token before reaching the 30 minutes. You can renew the token multiple times if you are using it. However, once the token reaches the 24 hours of its first creation, you can no longer renew the token.

Transits secret Engine

- can not store the encrypted data into encrypt endpoint however ciphertext can be stored in database
- . encryption key used to encrypt the plaintext is referred to as a data key
- Data key used to encrypt and decrypt large amount of data locally
- The data key is wrapped by the transits key
- Vault maintain the multiple version of the encryption key and Vault would refuse to decrypt the data if the key used is less than the minimum key version allowed
- secrets engine provides additional features (sign and verify data, generate hashes and HMACs of data, and act as a source of random bytes),

```
# Enable transit secrets engine
path "sys/mounts/transit" {
  capabilities = [ "create", "read", "update",
"delete", "list" ]
}

# To read enabled secrets engines
path "sys/mounts" {
  capabilities = [ "read" ]
}

# Manage the transit secrets engine
path "transit/*" {
  capabilities = [ "create", "read", "update",
"delete", "list" ]
}
```

Transit Policies

Create the token for the Policy

```
vault token create -policy=transit
```

Make request using generated token and encrypt the text in base64

```
VAULT_TOKEN=s.USYiT3KRz8uBJckyJEVrpH7g vault write transit/encrypt/orders \
➢ plaintext=$(base64 <<< "4111 1111 1111 1111")
```

```
Key            Value
---            -----
ciphertext     vault:v1:H+SWiFeKs/aNOMYwWEvhO47WTZSDQ5coL8bHliL48vAvRQy8HNntEWes+C53+STJ
key_version    1
```

Decrypt the text using same token

```
VAULT_TOKEN=s.USYiT3KRz8uBJckyJEVrpH7g vault write transit/decrypt/orders
ciphertext=vault:v1:
H+SWiFeKs/aNOMYwWEvhO47WTZSDQ5coL8bHliL48vAvRQy8HNntEWes+C53+STJ
```

```
Key            Value
---            -----
plaintext      NDExMSAxMTExIDExMTEgMTExMQo=
```

Decode the plaintext

base64 --decode <<< "NDExMSAxMTExIDExMTEgMTExMQo=“

4111 1111 1111 1111

Rotate the Encryption key

vault write -f transit/keys/orders/rotate

Encrypt the text again

VAULT_TOKEN=s.USYiT3KRz8uBJckyJEVrpH7g vault write transit/encrypt/orders plaintext=$(base64 <<< "4111 1111 1111 1111")

```
Key            Value
---            -----
ciphertext     vault:v2:PWjNV0IYexqLa8bQh3w34f+CYPo8X1yFUg1QfDJAchmyGp/MAaySRBCfqTAdK7Yq
key_version    2
```

Rewrap the key

VAULT_TOKEN=s.USYiT3KRz8uBJckyJEVrpH7g vault write transit/rewrap/orders ciphertext=vault:v1:H+SWiFeKs/aNOMYwWEvhO47WTZSDQ5coL8bHliL48vAvRQy8HNntEWes+C53+STJ

```
Key            Value
---            -----
ciphertext     vault:v2:G7vBTZo8QM2Pg7o6KUYSFoyx4SENOs7awlrLcGnIKn0hfYF3oum3huWzhZQtbV3l
key_version    2
```

Transit Secret Re-wrapping

Step Perform on the Vault Server

```
# Manage the transit secrets engine
path "transit/keys/*" {
  capabilities = [ "create", "read", "update", "delete", "list", "sudo" ]
}

# Enable the transit secrets engine
path "sys/mounts/transit" {
  capabilities = [ "create", "update" ]
}

# Write ACL policies
path "sys/policies/acl/*" {
  capabilities = [ "create", "read", "update", "delete", "list" ]
}

# Create tokens for verification & test
path "auth/token/create" {
  capabilities = [ "create", "update", "sudo" ]
}
```

Policy for the security engineer however it is optional if you used root user token

Transit Secret Re-wrapping

Step Perform on the Vault Server

step-1 vault secrets enable transit

step-2 vault write -f transit/keys/my_app_key

step-3 create the policy using app key

```
path "transit/keys/my_app_key" {
  capabilities = ["read"]
}

path "transit/rewrap/my_app_key" {
  capabilities = ["update"]
}

path "transit/encrypt/my_app_key" {
  capabilities = ["update"]
}
```

Transit Secret Re-wrapping

Step Perform on the Vault Server

step-4 vault policy write rewrap_example ./rewrap_example.hcl

Step-5 vault token create -policy=rewrap_example

Step-6 create another token APP_TOKEN=$(vault token create -format=json -policy=rewrap_example | jq -r ".auth.client_token")

Step-7 echo $APP_TOKEN [to display token]

Transit Secret Re-wrapping

Step Perform on the Application server

step -1 git clone https://github.com/hashicorp/vault-guides.git

Step -2 install the docker on the centos version 8

```
yum-config-manager \
 --add-repo \
https://download.docker.com/linux/centos/docker-ce.repo
```

Step -4 yum-config-manager --enable docker-ce-nightly

Step-5 yum install docker-ce docker-ce-cli containerd.io

Step-6 yum install dotnet-sdk-5.0

Transit Secret Re-wrapping

step-6 docker pull mysql/mysql-server:5.7

step-7
```
docker run --name mysql-rewrap \
    --publish 3306:3306 \
    --volume ~/rewrap-data:/var/lib/mysql \
    --env MYSQL_ROOT_PASSWORD=root \
    --env MYSQL_ROOT_HOST=% \
    --env MYSQL_DATABASE=my_app \
    --env MYSQL_USER=vault \
    --env MYSQL_PASSWORD=vaultpw \
    --detach mysql/mysql-server:5.7
```

Transit Secret Re-wrapping

step-8 VAULT_TOKEN=s.iZW56SyrmO7agdce8aXIIlpS \
VAULT_ADDR=https://vault-01.examsimulator.net:8200 \
VAULT_TRANSIT_KEY=my_app_key \
SHOULD_SEED_USERS=true \
dotnet run

```
[root@transit vault-transit-rewrap]# VAULT_TOKEN=s.iZW56SyrmO7agdce8aXIIlpS \
>       VAULT_ADDR=https://vault-01.examsimulator.net:8200 \
>       VAULT_TRANSIT_KEY=my_app_key \
>       SHOULD_SEED_USERS=true \
>       dotnet run
Connecting to Vault server...
Created (if not exist) my_app DB
Create (if not exist) user_data table
Seeded the database...
Moving rewrap...
Current Key Version: 1
Found 0 records to rewrap.
```

Transit Secret Re-wrapping

step-9 vault write -f transit/keys/my_app_key/rotate

step-10 Re-run the application

```
VAULT_TOKEN=s.iZW56SyrmO7agdce8aXIIlpS \
    VAULT_ADDR=https://vault-01.examsimulator.net:8200 \
    VAULT_TRANSIT_KEY=my_app_key \
    SHOULD_SEED_USERS=true \
    dotnet run
```

```
[root@transit vault-transit-rewrap]# VAULT_TOKEN=s.iZW56SyrmO7agdce8aXIIlpS \
>       VAULT_ADDR=https://vault-01.examsimulator.net:8200 \
>       VAULT_TRANSIT_KEY=my_app_key \
>       SHOULD_SEED_USERS=true \
>       dotnet run
Connecting to Vault server...
Created (if not exist) my_app DB
Create (if not exist) user_data table
Seeded the database...
Moving rewrap...
Current Key Version: 2
Found 500 records to rewrap.
Wrapped another 10 records: 10 so far...
Wrapped another 10 records: 20 so far...
Wrapped another 10 records: 30 so far...
Wrapped another 10 records: 40 so far...
Wrapped another 10 records: 50 so far...
Wrapped another 10 records: 60 so far...
Wrapped another 10 records: 70 so far...
Wrapped another 10 records: 80 so far...
Wrapped another 10 records: 90 so far...
Wrapped another 10 records: 100 so far...
```

Vault Agent

Auto-Auth - Automatically authenticate to Vault and manage the token renewal process for locally-retrieved dynamic secrets.

Caching - Allows client-side caching of responses containing newly created tokens and responses containing leased secrets generated off of these newly created tokens.

[Windows Service][winsvc] - Allows running the Vault Agent as a Windows service.
Templating - Allows rendering of user supplied templates by Vault Agent, using the token generated

Templating - Allows rendering of user supplied templates by Vault Agent, using the token generated by the Auto-Auth step. To get help, run:

Vault Agent Auto-Auth

Auto Auth is combination of auth method and Sink

Sinks, which are locations where the agent should write a token

Vault Agent caching

Responses containing new tokens will be cached by the agent only if the parent token is already being managed by the agent

to manage the stale cache agent used /agent/v1/cache-clear

Agent performs all operations in memory and does not persist anything to storage for the renewal of token

when the agent is shut down, all the renewal operations are immediately terminated and there is no way for agent to resume renewals after the fact.

Vault Agent Templates

The Vault Agent templating automatically renews and fetches secrets/tokens.

If a secret or token is renewable, Vault Agent will renew the secret after 2/3 of the secret's lease duration has elapsed.

If a secret or token isn't renewable or leased, Vault Agent will fetch the secret every 5 minutes

If a secret or token is non-renewable but leased, Vault Agent will fetch the secret when 85% of the secrets time-to-live (TTL) is reached

If a secret is a certificate, Vault Agent template will fetch the new certificate using the certificates validTo

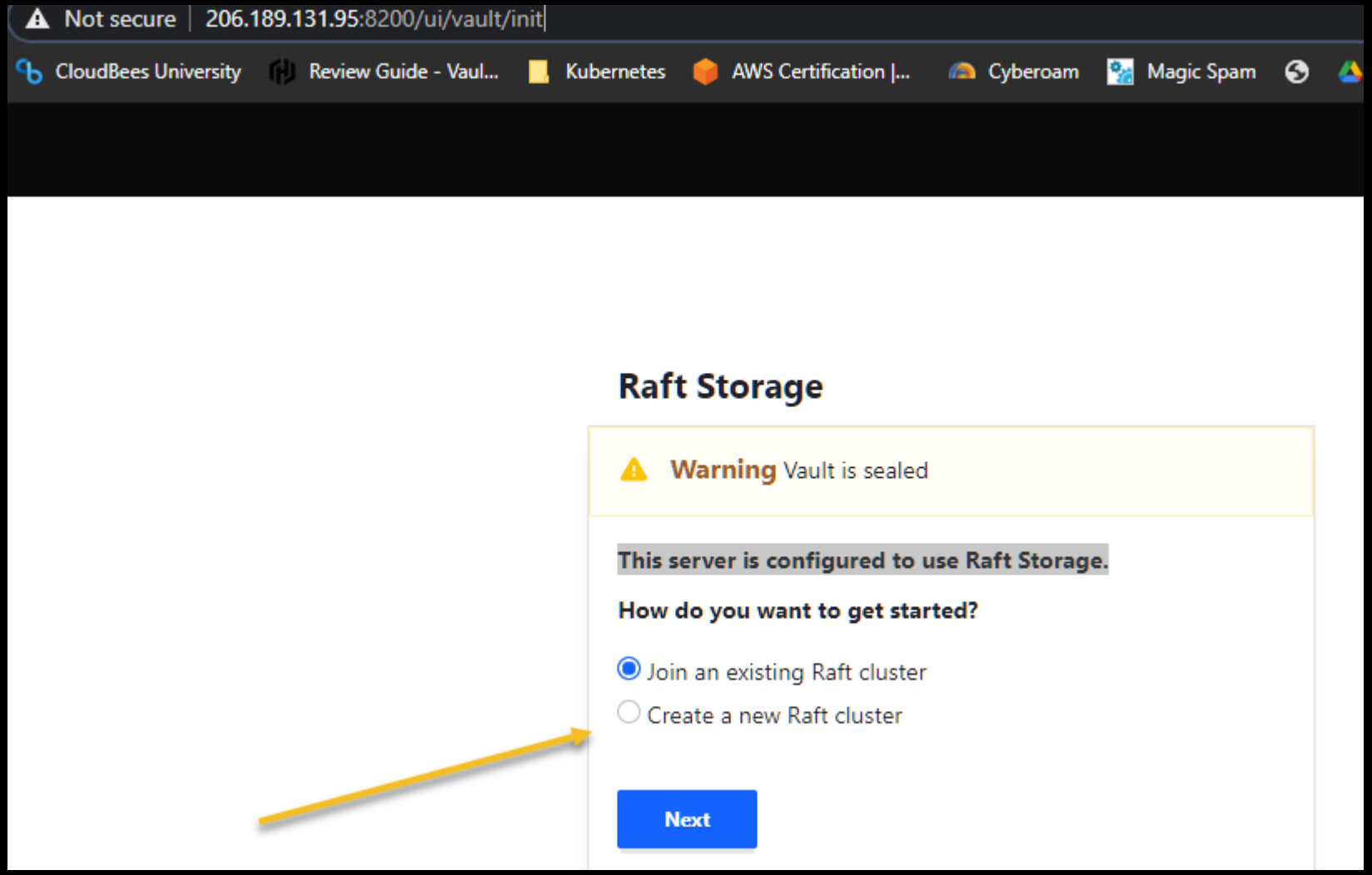
Vault in production mode

Vault configuration file

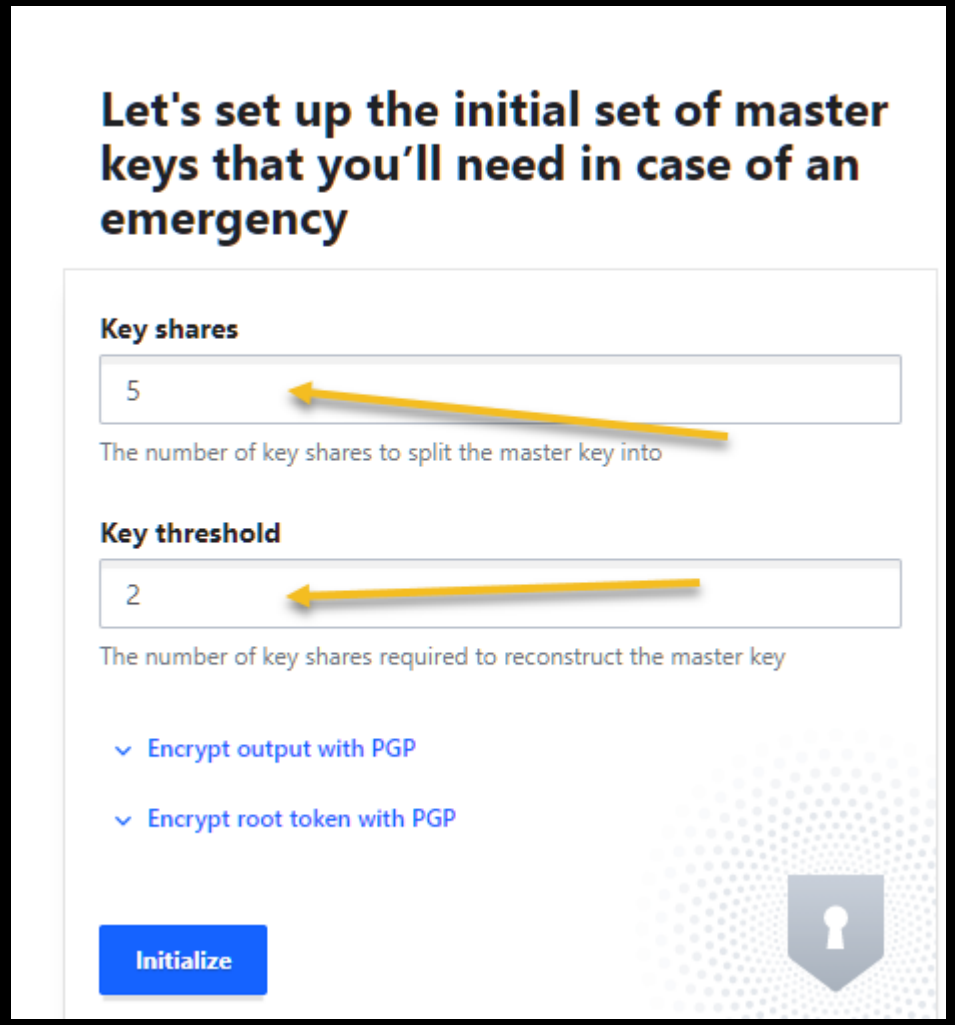
```
storage "raft" {
  path    = "./vault/data"
  node_id = "vault-server-01"
}

listener "tcp" {
  address     = "0.0.0.0:8200"
  tls_disable = "true"
}

api_addr = "http://127.0.0.1:8200"
cluster_addr = "https://127.0.0.1:8201"
ui = true
```

```
vault server -config=vault.hcl
```

Open the vault server in the web browser



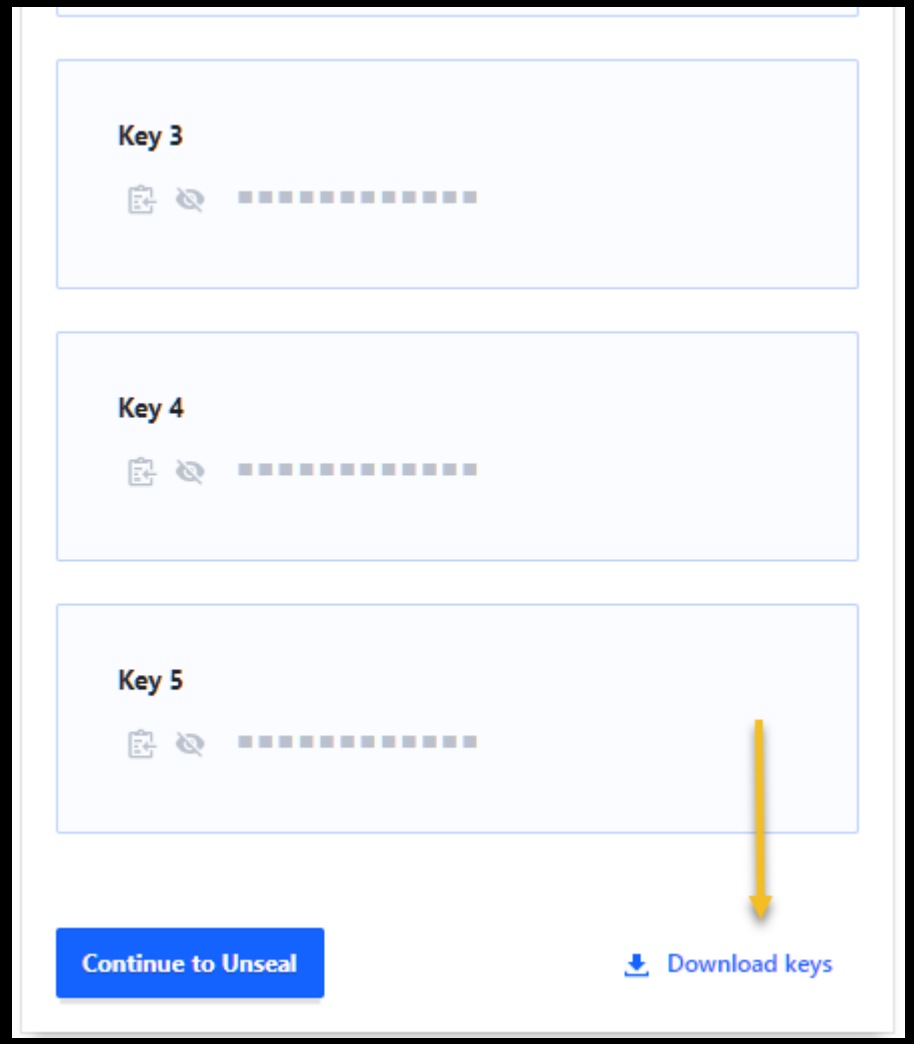
Open the vault server in the web browser



Total Key share is 5 but 2 key are required to unseal the vault server

This process also call initialization

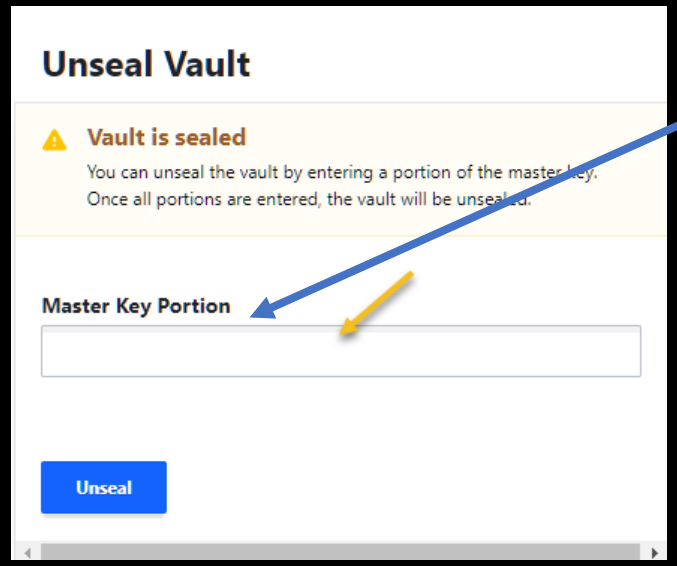
Download the key



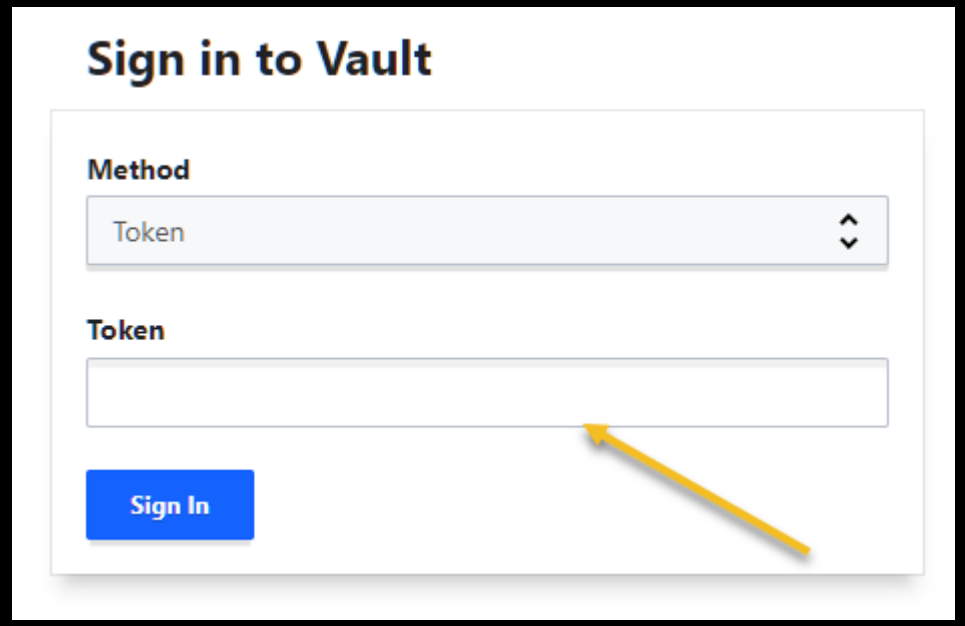
Open the vault server in the web browser

Insert any random key from it and unseal the vault


```
"keys": [
  "c6bc579e03c466dc10ba2737e02248eb8f53152833c8d8e7744bef8994641f6bb7",
  "7ab59ad592777d5e0c79b936c5c91923e6746fa6b447881eda12c556cf4a5c5fa2",
  "f32c3342d001cd24cb3e042786102fdf4515f4109be0afee7aac79d653b24306fc",
  "5738ec343006f157fbf9479c6df2de61d493f616895de524f6f2312160209a13d8",
  "c4f8d92b4edfaa78e04c73568a65509b6cfb1f3669b7a128fe769c8f6b888f2a03"
```

Insert the root token from the file



Vault server environment variable

```
[root@Vault-server-01 ~]# export VAULT_ADDR='http://127.0.0.1:8200'
[root@Vault-server-01 ~]# vault status
Key                     Value
---                     -----
Seal Type               shamir
Initialized             true
Sealed                  false
Total Shares            5
Threshold               2
Version                 1.7.0
Storage Type            raft
Cluster Name            vault-cluster-34ae6493
Cluster ID              f12bd8f8-835f-1064-7675-e54eef19d650
HA Enabled              true
HA Cluster              https://127.0.0.1:8201
HA Mode                 active
Active Since            2021-04-04T06:49:00.107933385Z
Raft Committed Index    36
Raft Applied Index      36
```

Vault operator unseal from the command line

```
[root@Vault-server-01 ~]# vault operator unseal
Unseal Key (will be hidden):
Key                     Value
---                     -----
Seal Type               shamir
Initialized             true
Sealed                  true
Total Shares            5
Threshold               2
Unseal Progress         1/2
Unseal Nonce            2ef7582e-a577-7f80-a854-b37c8c60a858
Version                 1.7.0
Storage Type            raft
HA Enabled              true
```

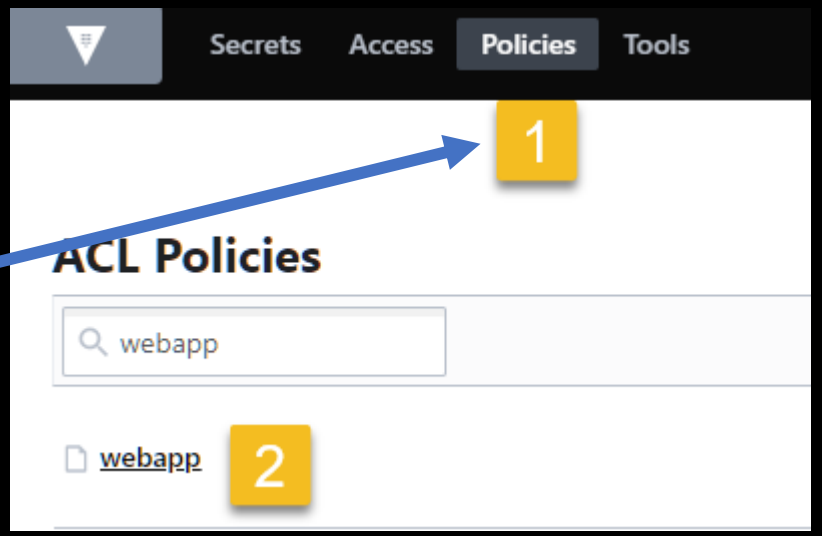
1

```
[root@Vault-server-01 ~]# vault operator unseal
Unseal Key (will be hidden):
Key                     Value
---                     -----
Seal Type               shamir
Initialized             true
Sealed                  false
Total Shares            5
Threshold               2
Version                 1.7.0
Storage Type            raft
Cluster Name            vault-cluster-34ae6493
Cluster ID              f12bd8f8-835f-1064-7675-e54eef19d650
HA Enabled              true
HA Cluster              n/a
HA Mode                 standby
Active Node Address     <none>
Raft Committed Index    44
Raft Applied Index      44
```

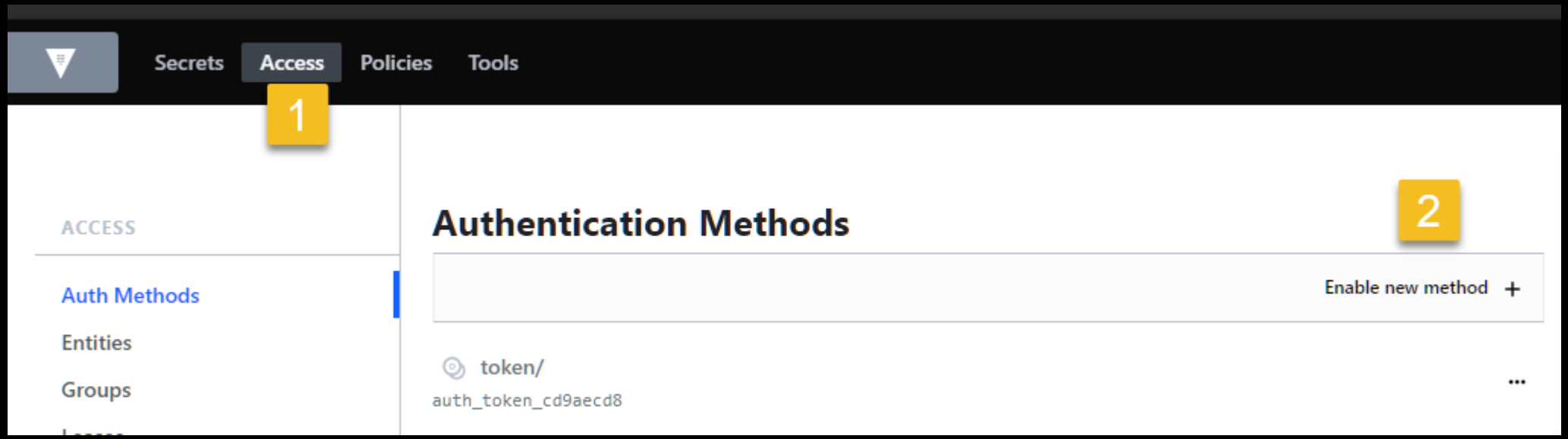
2

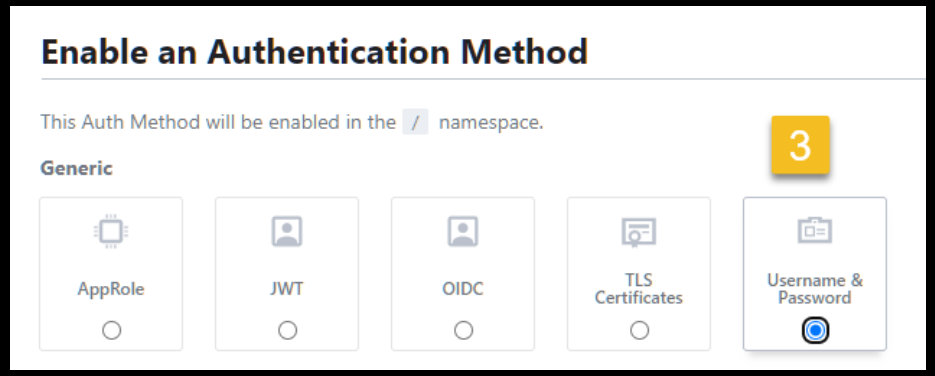
Userpass authentication method using ui

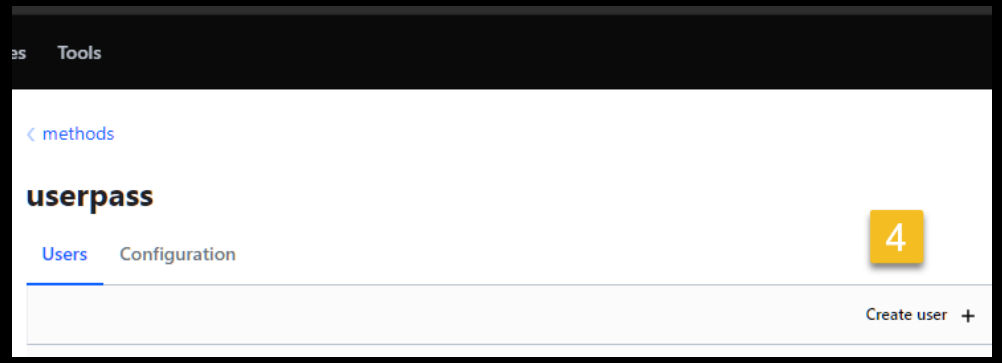
```
# Enable Transit secrets engine
path "sys/mounts/transit" {
   capabilities = ["create", "update"]
}

# Manage Transit secrets engine keys
path "transit/keys" {
   capabilities = ["list"]
}
path "transit/keys/*" {
   capabilities = ["create", "list", "read", "update"]
}
path "transit/keys/+/config" {
   capabilities = ["create", "update"]
}

# Encrypt with any Transit secrets engine key
path "transit/encrypt/*" {
   capabilities = ["create", "update"]
}

# Decrypt with any Transit secrets engine key
path "transit/decrypt/*" {
   capabilities = ["create", "update"]
}
```



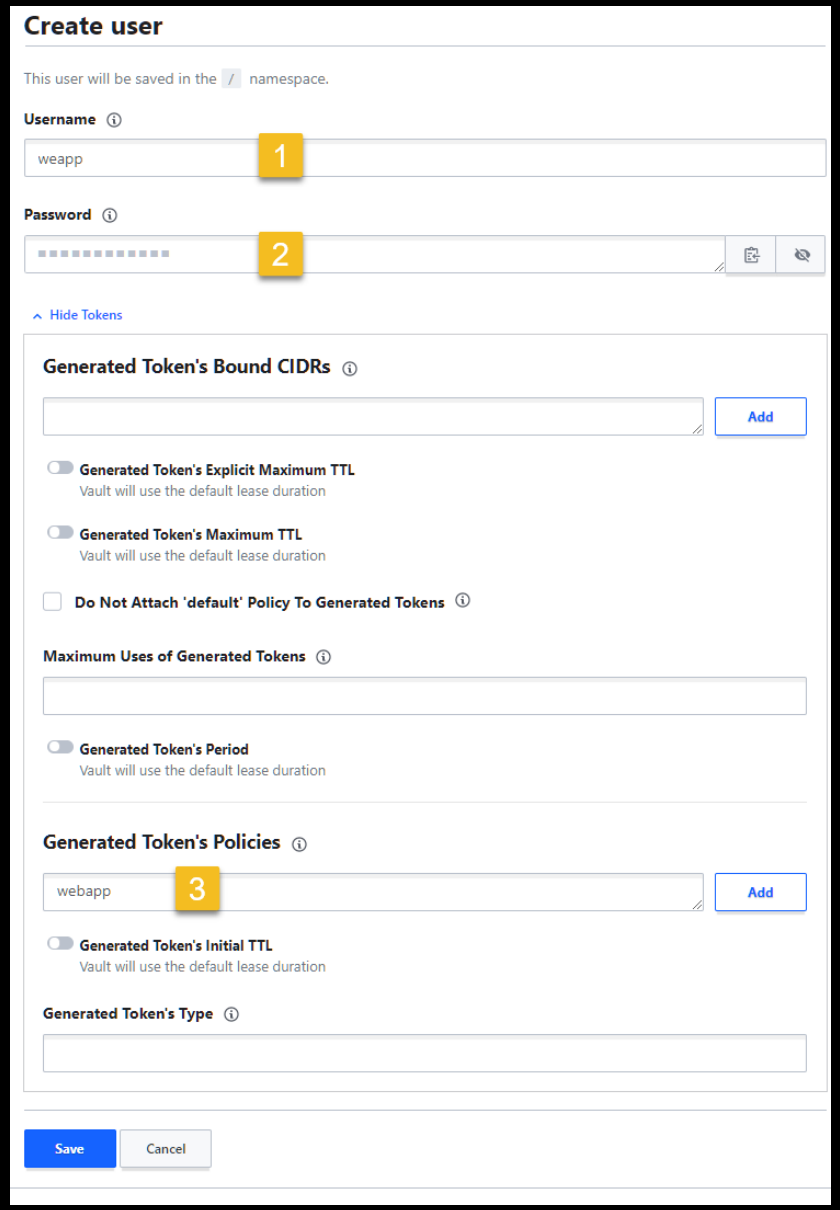
Userpass authentication method using ui

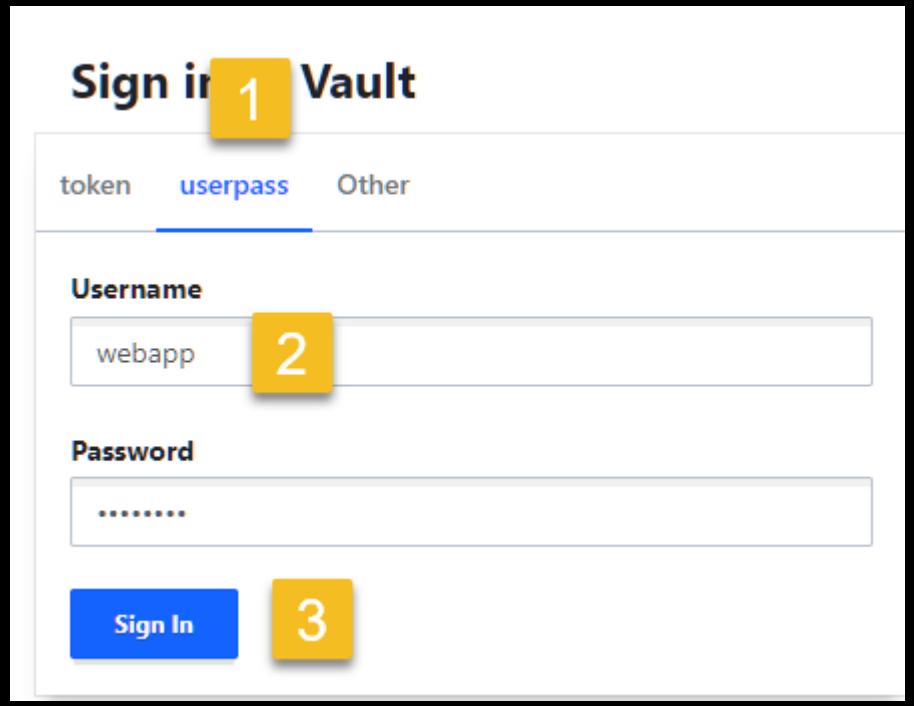






Userpass authentication method using ui





login with username and password which is created earlier using webapp Policy

Only allowed capabilities can be use d

Vault login with the pgp

Step-1 install the keybase application on the vault server from the keybase.io

Step-2 sudo yum install https://prerelease.keybase.io/keybase_amd64.rpm
run_keybase

Step- 3 start the vault server but using command line using vault server –config =vault.hcl

Step- 4 initialize the vault server for the pgp

```
vault operator init -key-shares=3 -key-threshold=2 \
>    -pgp-keys="keybase:jefferai,keybase:vishalnayak,keybase:sethvargo"
```

Generate the unseal key

```
[root@Vault-server-01 ~]# vault operator init -key-shares=3 -key-threshold=2 \
>     -pgp-key="keybase:jefferai,keybase:vishalnayak,keybase:sethvargo"
Unseal Key 1: 1 wMA37rwGt6FS1VAQgAUUa9eviDSHHhpbNRF1Dp1W+P+HvopIC6isjNFih0Mu/o9apicdX0eun6CaXiRCNcundrLhL6NKW9bppb2kWv5N3/hA9/A8kaOw+j
/8QbGViqnujSl[redacted]Oqfjr6hv9gfJx+Hs/2DdcUIOoJxuEwSnfsMHXHflQoO6pQ30a76xAqROCnVny5Q7aeBzlvyfM67xwAYKzHUd7778/2vC9b3l1Eef0bPfiBkXTjI0u3kfq
lJC9kFRDnqvIawiq4Q1FKfZcOOUfEYbsHrAgQZZQeWTDhsiTVmkJmh73IKFWWObsAJQ1pxSXwQ1qxTGRc+c5xJoXUDB00Va6HWAS0VuhF/9tLgAeQDE4ESB94LJkANXWTRF83x
lapN4J7gSeHtGeD/4p9Vs07g3+b9hdbAN2Y74ey5eQ2zaqJnTrDFFAp0jRrnDGbzETRyn5BeLzpxgfRd3L542a4Ilyk3y1/p1uhtQzuox3t1R7YH4MDh6RjgZORlZio9halcqC
CKYv8HKQ+m4sTA[redacted]4NYA
Unseal Key 2: 2 A0wwnMXgRzYYAQgAPGAtemgUyfA4LBaHbpQ2++pceCytukUoExl88Kn5i2dlytlUYb/3aoyF1CF6elKDr8HZQNjcr+XEMSBcSCY2zG7N8rYME1yowHDf
uuh26sxtMpHDU8[redacted]vhuoanWSlYZKbUGR1aSwq+NDvd83w6M+gcs0EtL3jIjI6FFD+ApVeXBnTiEsiG1fhSdyKjvlLrLEu8ZLwj0rGfksYSQ2Ta4AFefVa30H1NLlG65q8zkJ
jmNp2XylxeQnKkxpLTALpW4krHcNCHXCLoA9Hm+mrl/B4NToXfrgOXcyhsWWn2OVkl2++npmRKMRxN9cm3tV8/poKxWLI96riQVO2i9/NlAtLgAeSD4xB22BjM52ew68/oV1Ir
lSxy4CngJuFMBuBl4uFoNOPgRuYHDO8+q+uRj5sIRMwkbMwMMBb8/bCeTSU8XB5xSEiRw7XmsNkzMri+pgMkh3fTbgzkcvqG6dDzHD2T2c37Fbj64JHhU5bgvORzJSHkVKZXeW
/R80L3Y+5u4szb[redacted]TWAA
Unseal Key 3: 3 A0RVkFtoqzRlARAAbl1nZ0fOlRr4t89+g99YmK9kCBFWXeqktEhJHE16WC4K3eR4nRFOI8Y/OtZCeqJR8PsJngE47xdsfU5VgKVDfi2QK9qY6scvRH9e
PivDijoANGOvKr[redacted]GOLVOkwUY0q1PNIyri+tXKm4qb+AMdmrBW0/1w0zmTENkRZqOHcUVvT9XoBgtQRw726oUti/B3pdpWssVMmjnHYPCwOZBFPqEIeQBlLNaASmOoVFA8cY
v889/QOl2iom2FWZ3+kZkXMcjcUP+xuMKrAD8lzMQdngkmHB7gowCHoAONKBQJGgleQ9cMWC8ZPclvRHH2UhsxllcldxJ4cvVdhblOGTJFlPtubmvNOzHL3JHG7RFobfogLHr6
.i4FJJWzmCPmBdaI6tX2Ch6GG/sMV0f1hbCWhhmRYXOmJ/jXB5d7h0d9/WjtfyN71/WqQSc/nXoLAewGNMEG49U3mFRi0V4XjamsV3M454bfYLYkJMAd/c1sTAcwOLXm0QO2p7
5OyDGGI83CfT7PsnuPqa4wpiY5FKK1YAHFFwDfLNNFA6w/fjztpDj7DTS/yDHtWVpnBDctBePtfzBsQEITSrYgrzAHZQP4tOGI2FJbErsiQK6RIA9+j3Tu/I6kFwNaD5aI0dWq
R2ONSxXYA9ixqQTCvG5iclJHHuqlOKBI3pwalAR7TDyRe7XS4AHktDyckkLahL19cj6xpaEkLeFcmeCL4HjhoUTgIOIEOOCk4J/m7YuxZ0bmyFMiqOlmdT8e9MX2rbvswiE1hq
CRcx/X7oKQ05Cacr7DQ4qeKmuqUeGusIqKSRLg35YpRWTzu+ZebeBv4UhZ4GvkKQNGQqZZvR5z2N+gquJ3NuIc2J6b4YlKAA==

Initial Root Token: s.ZXQJqs7aKtKq9fBlbtuvt4iI 4

Vault initialized with 3 key shares and a key threshold of 2. Please securely
distribute the key shares printed above. When the Vault is re-sealed,
restarted, or stopped, you must supply at least 2 of these keys to unseal it
before it can start servicing requests.

Vault does not store the generated master key. Without at least 2 key to
reconstruct the master key, Vault will remain permanently sealed!

It is possible to generate new unseal keys, provided you have a quorum of
existing unseal keys shares. See "vault operator rekey" for more information.
```