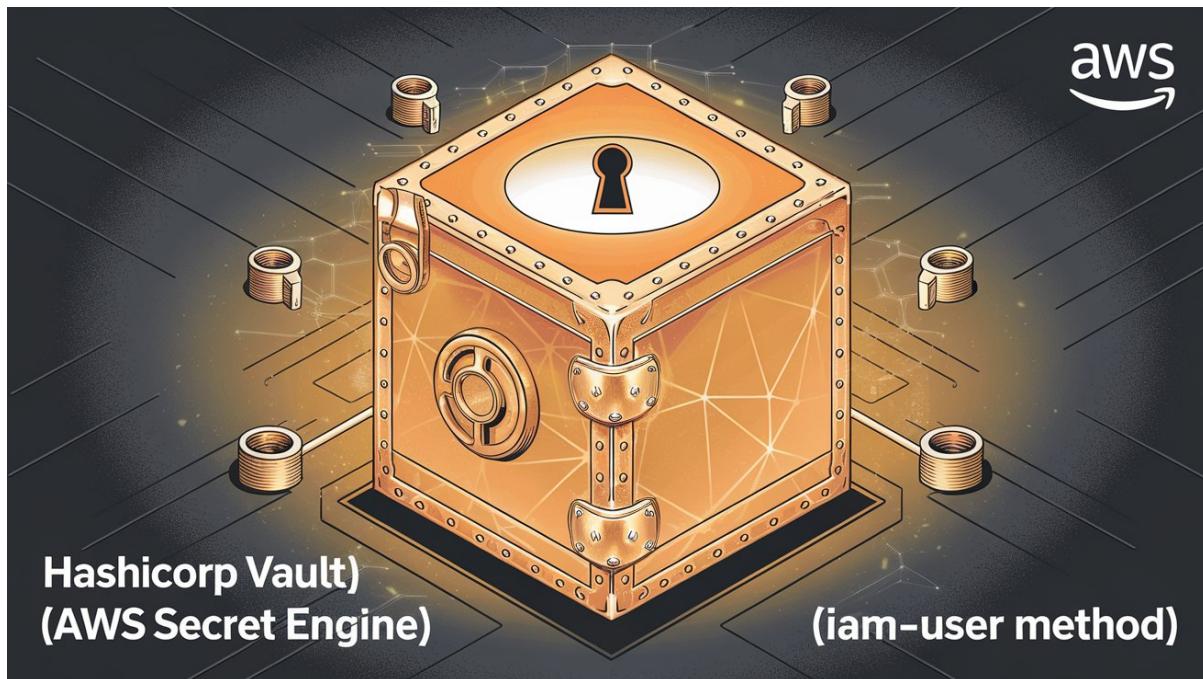


# Step-by-Step Guide to Configuring Dynamic AWS Credentials Using IAM\_User Method in HashiCorp Vault Using CLI



By

**Mahendran Selvakumar**

<https://devopstronaut.com/>



## Step1: Start Vault with Dev Server Mode

We can vault instance as dev mode using -dev

```
mahendranelvakumar@Mahendrans-MBP ~ % vault server -dev
==> Vault server configuration:

Administrative Namespace:
  Api Address: http://127.0.0.1:8200
  Cgo: disabled
  Cluster Address: https://127.0.0.1:8201
  Environment Variables: COLORFGBG, COLORTERM, COMMAND_MODE, HOME, HOMEBREW_CELLAR, HOMEBREW_PREFIX, HOMEBREW_REPOSITORY, INFOPATH, ITERM_PROFILE, ITERM_SESSION_ID, LANG, LC_TERMINAL, LC_TERMINAL_VERSION, LOGNAME, OLDPWD, PATH, PWD, SHELL, SHLVL, SSH_AUTH_SOCK, TERM, TERMIINFO_DIRS, TERM_FEATURES, TERM_PROGRAM, TERM_PROGRAM_VERSION, TERM_SESSION_ID, TMPDIR, USER, XPC_FLAGS, XPC_SERVICE_NAME, __CFBundleIdentifier, __CF_USE_R_TEXT_ENCODING
  Go Version: go1.22.7
  Listener 1: tcp (addr: "127.0.0.1:8200", cluster address: "127.0.0.1:8201", disable_request_limiter: "false", max_request_duration: "1m30s", max_request_size: "33554432", tls: "disabled")
  Log Level:
    Mlock: supported: false, enabled: false
  Recovery Mode: false
  Storage: inmem
  Version: Vault v1.18.0, built 2024-10-08T09:12:52Z
  Version Sha: 77f20b5610a4b6b1cc5071b8624cefef7a72e84

==> Vault server started! Log data will stream in below:
```

```
2024-10-27T22:27:49.388Z [INFO] core: successful mount: namespace="" path=secret/ type=kv version="v0.20.0+builtin"
WARNING! dev mode is enabled! In this mode, Vault runs entirely in-memory
and starts unsealed with a single unseal key. The root token is already
authenticated to the CLI, so you can immediately begin using Vault.
```

You may need to set the following environment variables:

```
$ export VAULT_ADDR='http://127.0.0.1:8200'
```

The unseal key and root token are displayed below in case you want to
seal/unseal the Vault or re-authenticate.

```
Unseal Key: Tl1HaNqU6abkhpH90wnYFcI91nTbBi4QW0eguca/1L8=
Root Token: hvs.R1xwf0oqTh4iPD5F9l0LmJEK
```

Development mode should NOT be used in production installations!

```
2024-10-27T22:30:35.669Z [INFO] core: successful mount: namespace="" path=aws/ type=aws version="v1.18.0+builtin.vault"
```

Vault server started as dev mode and it's running with in-memory if we close the terminal we can't able to access the vault.

## Set Environment Variable

Open new terminal of Vault server and set environment variable and Verify Vault Setup and Access Secrets

```
Last login: Sun Oct 27 22:27:03 on ttys001
mahendranelvakumar@Mahendrans-MBP ~ % export VAULT_ADDR='http://127.0.0.1:8200'
mahendranelvakumar@Mahendrans-MBP ~ % vault status
Key          Value
---          ---
Seal Type    shamir
Initialized   true
Sealed       false
Total Shares 1
Threshold    1
Version      1.18.0
Build Date   2024-10-08T09:12:52Z
Storage Type inmem
Cluster Name vault-cluster-090a0f79
Cluster ID   51ce616b-0db4-1713-5384-951ab22c5e6b
HA Enabled   false
mahendranelvakumar@Mahendrans-MBP ~ %
```



Now we can see Vault server seal type as Shamir and it's initialized, unsealed automatically. Vault storage type also in memory so we can't use in production.

## Step2: Enable AWS Secrets Engine using the CLI

Enable the AWS secrets engine using **vault secrets enable aws** command

```
mahendralselvakumar@Mahendrans-MBP ~ % vault secrets enable aws
Success! Enabled the aws secrets engine at: aws/
mahendralselvakumar@Mahendrans-MBP ~ %
```

To list Vault secrets, use the **vault secrets list** command

```
mahendralselvakumar@Mahendrans-MBP ~ % vault secrets list
Path          Type      Accessor        Description
----          ----      -----        -----
aws/          aws       aws_6975c27a   n/a
cubbyhole/    cubbyhole cubbyhole_86589c0d per-token private secret storage
identity/    identity  identity_2c60f021 identity store
secret/       kv        kv_5c4cb679   key/value secret storage
sys/          system    system_76dad5d4  system endpoints used for control, policy and debugging
mahendralselvakumar@Mahendrans-MBP ~ %
```

## Step3: Create an IAM user to integrate with Vault and provide the necessary permissions

### 3a.Create an IAM policy to manage dynamic IAM user credentials

The screenshot shows the AWS IAM Dashboard. On the left, there's a sidebar with 'Identity and Access Management (IAM)' selected. Under 'Access management', 'Policies' is highlighted. The main area displays 'Security recommendations' with two items: 'Root user has MFA' and 'Root user has no active access keys'. To the right, there's a 'AWS Account' panel showing 'Account ID: 038462791702' and a 'Quick Links' panel.

Navigate to **Policies** and click **Create Policy**

The screenshot shows the 'Policies' page with 1243 policies listed. The left sidebar has 'Policies' selected under 'Access management'. The main table lists policies like 'AccessAnalyzerService...', 'AdministratorAccess...', and 'AdministratorAccess....'. A 'Create policy' button is visible at the top right of the table.



Switch the policy editor to JSON mode, and copy the IAM permission policy from the HashiCorp website (<https://developer.hashicorp.com/vault/docs/secrets/aws#iam-permissions-policy-for-vault>)

IAM > Policies > Create policy

Step 1  
Specify permissions

Step 2  
Review and create

Policy editor

Visual JSON Actions ▾

▼ Select a service

Specify what actions can be performed on specific resources in a service.

Service

Choose a service

+ Add more permissions

Cancel Next

Modify the account number in the **Resource** section of the policy to match your AWS account number

Step 1  
Specify permissions

Step 2  
Review and create

Policy editor

Visual JSON Actions ▾

```
1 {  
2   "Version": "2012-10-17",  
3   "Statement": [  
4     {  
5       "Effect": "Allow",  
6       "Action": [  
7         "iam:AttachUserPolicy",  
8         "iam:CreateAccessKey",  
9         "iam:CreateUser",  
10        "iam:DeleteAccessKey",  
11        "iam:DeleteUser",  
12        "iam:DeleteUserPolicy",  
13        "iam:DetachUserPolicy",  
14        "iam:GetUser",  
15        "iam>ListAccessKeys",  
16        "iam>ListAttachedUserPolicies",  
17        "iam>ListGroupsForUser",  
18        "iam>ListUserPolicies",  
19        "iam:PutUserPolicy",  
20        "iam:AddUserToGroup",  
21        "iam:RemoveUserFromGroup"  
22      ],  
23      "Resource": ["arn:aws:iam::038462791702:user/vault-*"]  
24    }  
25  ]  
26}  
27
```

Edit statement

Select a statement

Select an existing statement in the policy or add a new statement.

+ Add new statement

Enter a **policy name** and click **Create Policy**



**Policy details**

**Policy name**  
Enter a meaningful name to identify this policy.  
 Maximum 128 characters. Use alphanumeric and "+-, @-\_" characters.

**Description - optional**  
Add a short explanation for this policy.  
 Maximum 1,000 characters. Use alphanumeric and "+-, @-\_" characters.

ⓘ This policy defines some actions, resources, or conditions that do not provide permissions. To grant access, policies must have an action that has an applicable resource or condition. For details, choose Show remaining. [Learn more](#)

**Permissions defined in this policy** Info Edit

Permissions defined in this policy document specify which actions are allowed or denied. To define permissions for an IAM identity (user, user group, or role), attach a policy to it.

Show remaining 422 services

**Allow (1 of 423 services)**

Service	Access level	Resource	Request condition
IAM	Limited: List, Permissions management, Read, Write	UserName  string like  vault-*	None

**Add tags - optional** Info  
Tags are key-value pairs that you can add to AWS resources to help identify, organize, or search for resources.  
No tags associated with the resource.  
 You can add up to 50 more tags.

Cancel Previous **Create policy**

## The Policy has been created

Policy vault-dynamic-creds created.		<a href="#">View policy</a>	X
<a href="#">IAM</a>	> Policies		
<b>Policies (1244) <small>Info</small></b> A policy is an object in AWS that defines permissions.			
<input type="button" value="C"/> Actions <input type="button" value="Delete"/> <b>Create policy</b>	<input type="text" value="Search"/> Filter by Type <input type="button" value="All types"/> <input type="button" value="Description"/>	<input type="button" value="1"/> 2 3 4 5 6 7 ... 63 > <input type="button" value="Reset"/>	
Policy name	Type	Used as	Description
<input type="radio"/> <a href="#">AccessAnalyzerServiceRolePolicy</a>	AWS managed	None	Allow Access Analyzer to analyze resou...

## 3b.Create an IAM user and assign the newly created IAM policy to the user

Go to **Users** in IAM and click **Create user**

Identity and Access Management (IAM)		<a href="#">IAM</a> > Users
<input type="text" value="Search IAM"/>		<input type="button" value="C"/> Delete <b>Create user</b>
<b>Users (0) <small>Info</small></b> An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.		
<input type="text" value="Search"/>	<input type="button" value="User name"/> Path Group: Last activity MFA Password age Console last sign-in Access key ID Active key age Access	<input type="button" value="1"/> > <input type="button" value="Reset"/>

Enter the user name and click “Next”

<https://www.linkedin.com/in/mahendran-selvakumar-36444a77/>



IAM > Users > Create user

Step 1 Specify user details

Step 2 Set permissions

Step 3 Review and create

### Specify user details

User details

User name  The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and + = . \_ - (hyphen)

Provide user access to the AWS Management Console - optional If you're providing console access to a person, it's a best practice to manage their access in IAM Identity Center.

If you are creating programmatic access through access keys or service-specific credentials for AWS CodeCommit or Amazon Keyspaces, you can generate them after you create this IAM user. [Learn more](#)

[Cancel](#) [Next](#)

Select **Attach policies directly**, choose the policy you created, and click **Next**

IAM > Users > Create user

Step 1 Specify user details

Step 2 Set permissions

Step 3 Review and create

### Set permissions

Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. [Learn more](#)

Add user to group Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.

Copy permissions Copy all group memberships, attached managed policies, and inline policies from an existing user.

Attach policies directly Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.

**Permissions policies (1/1246)**  
Choose one or more policies to attach to your new user.

Policy name	Type	Attached entities
<input checked="" type="checkbox"/> vault-dynamic-creds	Customer managed	0

**Set permissions boundary - optional**

[Cancel](#) [Previous](#) [Next](#)

Review and click Create user

### Review and create

Review your choices. After you create the user, you can view and download the autogenerated password, if enabled.

**User details**

User name vault-user	Console password type None	Require password reset No
-------------------------	-------------------------------	------------------------------

**Permissions summary**

Name	Type	Used as
vault-dynamic-creds	Customer managed	Permissions policy

**Tags - optional**  
Tags are key-value pairs you can add to AWS resources to help identify, organize, or search for resources. Choose any tags you want to associate with this user.

No tags associated with the resource.

[Add new tag](#)  
You can add up to 50 more tags.

[Cancel](#) [Previous](#) [Create user](#)



### 3c. Create Programmatic access to the user

Go to the created user and click Create access key

IAM > Users > vault-user

vault-user [Info](#) [Delete](#)

**Summary**

ARN arn:aws:iam::038462791702:user/vault-user	Console access Disabled	Access key 1 <a href="#">Create access key</a>
Created October 27, 2024, 22:49 (UTC)	Last console sign-in -	

Permissions Groups Tags **Security credentials** Last Accessed

**Console sign-in** [Enable console access](#)

Console sign-in link <a href="https://038462791702.sigin.aws.amazon.com/console">https://038462791702.sigin.aws.amazon.com/console</a>	Console password Not enabled
---	---------------------------------

**Multi-factor authentication (MFA) (0)** [Remove](#) [Resync](#) [Assign MFA device](#)

Type	Identifier	Certifications	Created on
No MFA devices. Assign an MFA device to improve the security of your AWS environment			
<a href="#">Assign MFA device</a>			

**Access keys (0)** [Create access key](#)

No access keys. As a best practice, avoid using long-term credentials like access keys. Instead, use tools which provide short term credentials. <a href="#">Learn more</a>
<a href="#">Create access key</a>

Select **Third-party service**, check the box for **I understand**, and click **Next**

IAM > Users > vault-user > Create access key

Step 1 **Access key best practices & alternatives** [Info](#)

Avoid using long-term credentials like access keys to improve your security. Consider the following use cases and alternatives.

Step 2 - optional  
Set description tag

Step 3  
Retrieve access keys

**Use case**

- Command Line Interface (CLI)**  
You plan to use this access key to enable the AWS CLI to access your AWS account.
- Local code**  
You plan to use this access key to enable application code in a local development environment to access your AWS account.
- Application running on an AWS compute service**  
You plan to use this access key to enable application code running on an AWS compute service like Amazon EC2, Amazon ECS, or AWS Lambda to access your AWS account.
- Third-party service**  
You plan to use this access key to enable access for a third-party application or service that monitors or manages your AWS resources.
- Application running outside AWS**  
You plan to use this access key to authenticate workloads running in your data center or other infrastructure outside of AWS that needs to access your AWS resources.
- Other**  
Your use case is not listed here.

**Alternative recommended**  
As a best practice, use temporary security credentials (IAM roles) instead of creating long-term credentials like access keys, and don't create AWS account root user access keys. [Learn more](#)

**Confirmation**  
 I understand the above recommendation and want to proceed to create an access key.

[Cancel](#) **Next**



Provide the **description tag value** and click **Create access key**

IAM > Users > vault-user > Create access key

Step 1  
[Access key best practices & alternatives](#)

Step 2 - optional  
**Set description tag**

Step 3  
Retrieve access keys

**Set description tag - optional** Info

The description for this access key will be attached to this user as a tag and shown alongside the access key.

**Description tag value**

Describe the purpose of this access key and where it will be used. A good description will help you rotate this access key confidently later.

Maximum 256 characters. Allowed characters are letters, numbers, spaces representable in UTF-8, and: \_ . : / = + - @

[Cancel](#) [Previous](#) [Create access key](#)

The access key has been created, and we can view or download the secret access key only at this time; it cannot be recovered later

**Access key created**

This is the only time that the secret access key can be viewed or downloaded. You cannot recover it later. However, you can create a new access key any time.

IAM > Users > vault-user > Create access key

Step 1  
[Access key best practices & alternatives](#)

Step 2 - optional  
[Set description tag](#)

Step 3  
**Retrieve access keys**

**Retrieve access keys** Info

**Access key**

If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.

Access key	Secret access key
<input type="checkbox"/> AKIAQR5EPUALKM6GV6XB	<input type="checkbox"/> ***** <a href="#">Show</a>

**Access key best practices**

- Never store your access key in plain text, in a code repository, or in code.
- Disable or delete access key when no longer needed.
- Enable least-privilege permissions.
- Rotate access keys regularly.

For more details about managing access keys, see the [best practices for managing AWS access keys](#).

[Download .csv file](#) [Done](#)



## Step 4: Configure the credentials that Vault uses to communicate with AWS

Run the vault write command to configure the credentials

```
vault write aws/config/root \
access_key=AKIAQR5EPUALKM6GV6XB \
secret_key=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 \
region=eu-west-1
```

```
mahendralselvakumar@Mahendrans-MBP ~ % vault write aws/config/root \
access_key=AKIAQR5EPUALKM6GV6XB \
secret_key=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 \
region=eu-west-1
Success! Data written to: aws/config/root
mahendralselvakumar@Mahendrans-MBP ~ %
```

Run the command **vault read aws/config/root** to retrieve the current configuration for the AWS secrets engine

```
mahendralselvakumar@Mahendrans-MBP ~ % vault read aws/config/root
Key          Value
---          ---
access_key    AKIAQR5EPUALKM6GV6XB
iam_endpoint  n/a
identity_token_audience  n/a
identity_token_ttl  0s
max_retries   -1
region        eu-west-1
role_arn      n/a
sts_endpoint  n/a
username_template {{ if (eq (.Type "STS") ){{ printf "vault-%s-%s" (unix_time) (random 20) | truncate 32 }}{{ else }}{{ printf "vault-%s-%s-%s" (printf "%s-%s" (.DisplayName) (.PolicyName) | truncate 42) (unix_time) (random 20) | truncate 64 }}{{ end }}}
mahendralselvakumar@Mahendrans-MBP ~ %
```

Rotate the AWS Credentials by using the **vault write -f aws/config/rotate-root**

```
mahendralselvakumar@Mahendrans-MBP ~ % vault write -f aws/config/rotate-root
Key          Value
---          ---
access_key    AKIAQR5EPUALDI3EQMMN
mahendralselvakumar@Mahendrans-MBP ~ %
```

Run the command **vault read aws/config/root** to view the rotated access key

```
mahendralselvakumar@Mahendrans-MBP ~ % vault read aws/config/root
Key          Value
---          ---
access_key    AKIAQR5EPUALDI3EQMMN
iam_endpoint  n/a
identity_token_audience  n/a
identity_token_ttl  0s
max_retries   -1
region        eu-west-1
role_arn      n/a
sts_endpoint  n/a
username_template {{ if (eq (.Type "STS") ){{ printf "vault-%s-%s" (unix_time) (random 20) | truncate 32 }}{{ else }}{{ printf "vault-%s-%s-%s" (printf "%s-%s" (.DisplayName) (.PolicyName) | truncate 42) (unix_time) (random 20) | truncate 64 }}{{ end }}}
mahendralselvakumar@Mahendrans-MBP ~ %
```



If you check in the AWS console, a new secret key has been created, and the old access key has been deleted

The screenshot shows the AWS IAM Access Keys page. At the top, there is a header with the text "Access keys (1)" and a "Create access key" button. Below the header, there is a table with one row. The row contains the following information:

AKIAQR5EPUALD13EQMMN	Description	Status Active	Actions ▾
-	Last used	None	
N/A	Last used region	Created 1 minute ago	
N/A	Last used service	N/A	

## Step 5: Create an IAM Role to Manage AWS IAM User Credentials

To create a new role in Vault for managing AWS IAM user credentials, use the following command

```
vault write aws/roles/vault-role \
```

```
policy_arns=arn:aws:iam::aws:policy/AmazonEC2ReadOnlyAccess \
```

```
credential_type=iam_user
```

Here, I have used **AmazonEC2ReadOnlyAccess**, but you can use any role

```
mahendralselvakumar@Mahendrals-MBP ~ % vault write aws/roles/vault-role \  
> policy_arns=arn:aws:iam::aws:policy/AmazonEC2ReadOnlyAccess \  
> credential_type=iam_user  
Success! Data written to: aws/roles/vault-role  
mahendralselvakumar@Mahendrals-MBP ~ %
```

Run the command **vault read aws/creds/vault-role** to generate and retrieve temporary AWS IAM user credentials associated with the vault-role

```
mahendralselvakumar@Mahendrals-MBP ~ % vault read aws/creds/vault-role  
Key          Value  
---          ----  
lease_id      aws/creds/vault-role/mPohdODWenFBaTlb0MeeHYxn  
lease_duration 768h  
lease_renewable true  
access_key    AKIAQR5EPUALLUEPMJEF  
secret_key    UpTJS0gYcdU1jCdXxiXQBFevnMXX/raKCMqCHjb  
session_token <nil>  
mahendralselvakumar@Mahendrals-MBP ~ %
```

A new IAM user is created when you run the command **vault read aws/creds/vault-role**



IAM > Users

Users (2) Info

An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.

User name	Path	Groups	Last activity	MFA	Password age	Console last sign-in	Access key ID	Active key age	Access key
vault-root-vault-role-1730070842-UovDzclD7JWk2vZcYIUG	/	0	-	-	-	-	-	-	-
vault-user	/	0	-	-	-	-	Active - AKIAQR5EPUA...	12 minutes	-

If you run this command multiple times, multiple users will be created, each with the default lease duration.

```
mahendralselvakumar@Mahendrans-MBP ~ % vault read aws/creds/vault-role
Key          Value
---          ---
lease_id     aws/creds/vault-role/CVtWVvp5Fpa0FD6o3kKKJ7Te
lease_duration 768h
lease_renewable true
access_key   AKIAQR5EPUALDQHRT7WX
secret_key    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
session_token <nil>
mahendralselvakumar@Mahendrans-MBP ~ % vault read aws/creds/vault-role
Key          Value
---          ---
lease_id     aws/creds/vault-role/Y2GapAxEv08kh4gX5ce2WlsR
lease_duration 768h
lease_renewable true
access_key   AKIAQR5EPUALIXKWN4CB
secret_key    kv+/VrAFIU SudFUJ/6s9rI42NMyKhIXQC9QxjyUW
session_token <nil>
mahendralselvakumar@Mahendrans-MBP ~ %
```

Multiple IAM users were created because we ran the **vault read aws/creds/vault-role** command multiple times. We can use the access key and secret key from these users to configure the AWS CLI.

IAM > Users

Users (4) Info

An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.

User name	Path	Groups	Last activity	MFA	Password age	Console last sign-in	Access key ID	Active key age	Access key
vault-root-vault-role-1730070842-UovDzclD7JWk2vZcYIUG	/	0	-	-	-	-	-	-	-
vault-root-vault-role-1730071019-2CRYquGAB22ZGbMvkJpl	/	0	-	-	-	-	-	-	-
vault-root-vault-role-1730071023-jyPVC4PP37ab9Ejs0Rf	/	0	-	-	-	-	-	-	-
vault-user	/	0	-	-	-	-	Active - AKIAQR5EPUA...	15 minutes	-



## Revoke Vault credential:

Revoke a particular lease using this command **vault lease revoke aws/creds/vault-role/Y2GapAxEvO8kh4gX5ce2WlsR**

```
mahendralselvakumar@Mahendrans-MBP ~ % vault lease revoke aws/creds/vault-role/Y2GapAxEvO8kh4gX5ce2WlsR
All revocation operations queued successfully!
mahendralselvakumar@Mahendrans-MBP ~ %
```

**vault lease revoke:** This command is used to revoke a specific lease in Vault

**aws/creds/vault-role/Y2GapAxEvO8kh4gX5ce2WlsR:** This specifies the path to the lease you want to revoke. The string Y2GapAxEvO8kh4gX5ce2WlsR is the lease ID for the IAM user credentials that you wish to revoke

We can also revoke leases with a prefix using the command **vault lease revoke -prefix aws/creds/vault-role**

```
VERSION HISTORY      View the version history of the target Vault server
mahendralselvakumar@Mahendrans-MBP ~ % vault lease revoke -prefix aws/creds/vault-role
All revocation operations queued successfully!
mahendralselvakumar@Mahendrans-MBP ~ %
```

Now all the users have been deleted

User name	Path	Group	Last activity	MFA	Password age	Console last sign-in	Access key ID	Active key age	Access key
vault-user	/	0	5 minutes ago	-	-	-	Active - AKIAQRSEPUA...	21 minutes	5 min

## Conclusion

In this guide, we successfully configured HashiCorp Vault to manage dynamic AWS credentials using the IAM User method through the CLI. By enabling the AWS Secrets Engine via command-line operations, we created a dedicated role and generated temporary IAM user credentials. These credentials provide secure, short-lived access to AWS resources, reducing the risk of credential leakage and enhancing security.

It's important to note that once generated, these temporary credentials cannot be accessed later, ensuring that sensitive information remains protected. This dynamic approach allows for efficient management of access permissions in a cloud environment, automating the credential lifecycle while maintaining strict security controls, all executed seamlessly through the CLI.



Keep Learning, Keep Securing!!

Feel free to reach out to me, if you have any other queries or suggestions Stay connected on LinkedIn: [Mahendran Selvakumar](https://www.linkedin.com/in/mahendran-selvakumar-36444a77/)

Stay connected on Medium: <https://devopstronaut.com/>