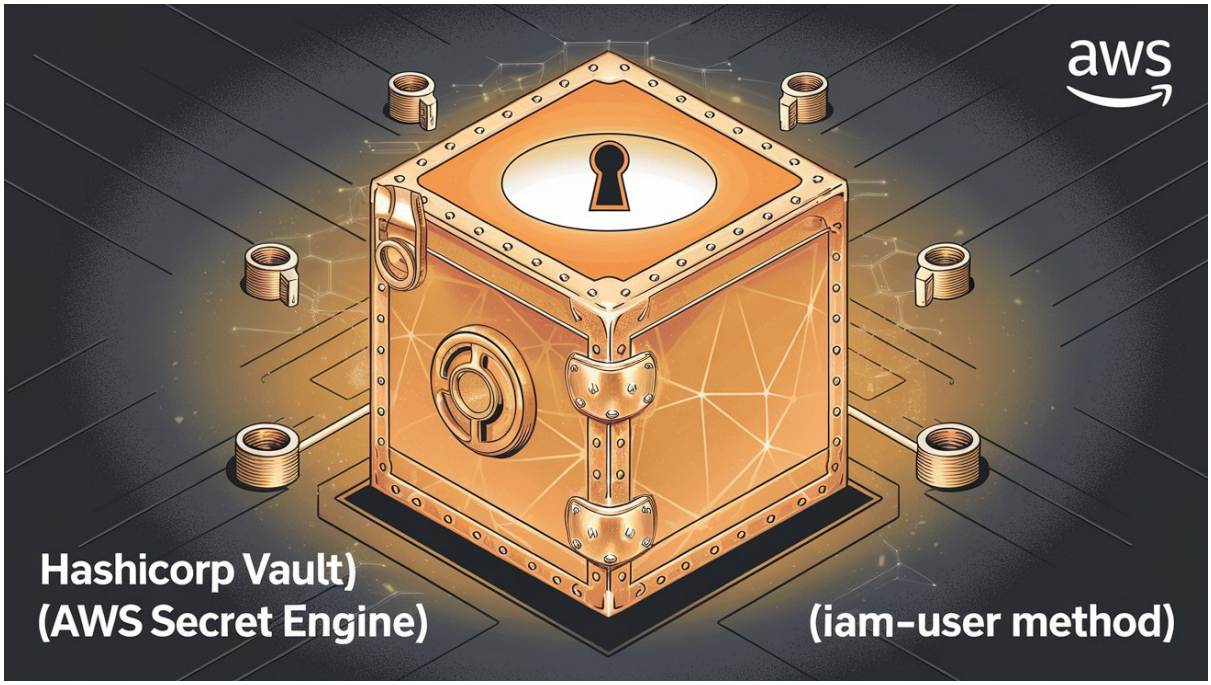
Step-by-Step Guide to Configuring Dynamic AWS Credentials Using IAM_User Method in HashiCorp Vault Using CLI



By

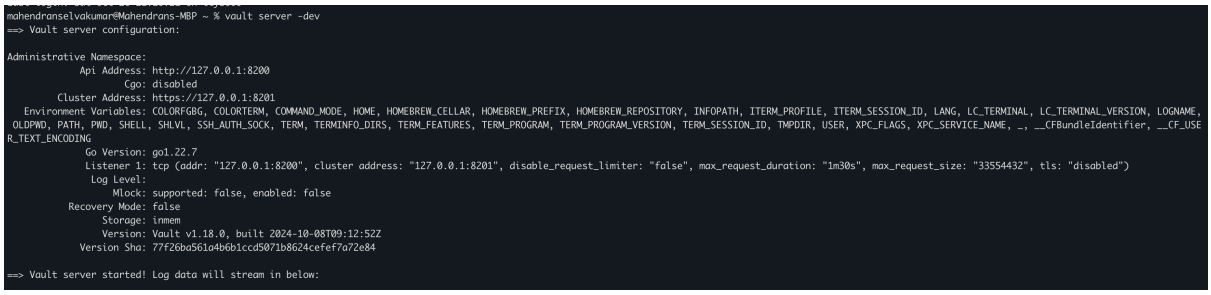
Mahendran Selvakumar

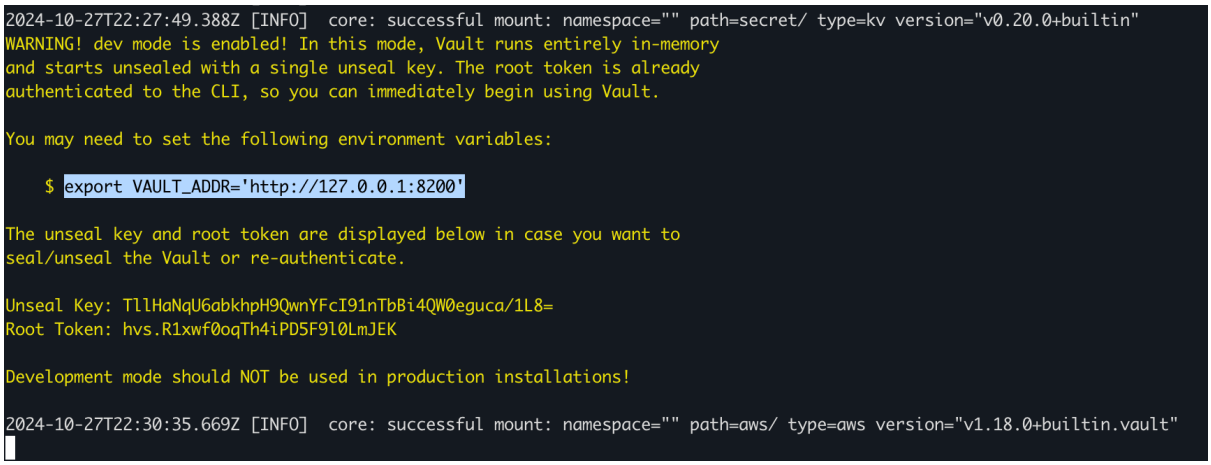
https://devopstronaut.com/

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

Step1: Start Vault with Dev Server Mode

We can vault instance as dev mode using *-dev*





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

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

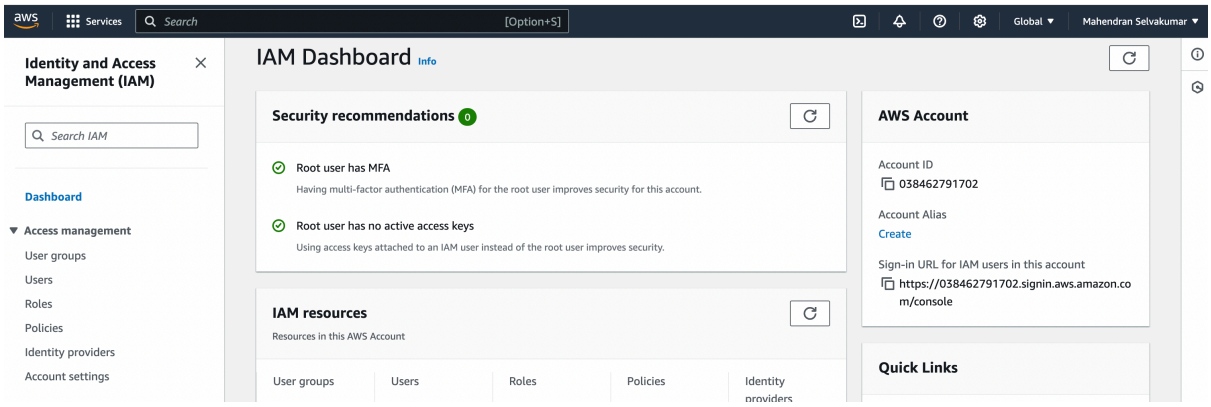
```
mahendranselvakumar@Mahendrans-MBP ~ % vault secrets enable aws
Success! Enabled the aws secrets engine at: aws/
mahendranselvakumar@Mahendrans-MBP ~ %
```

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

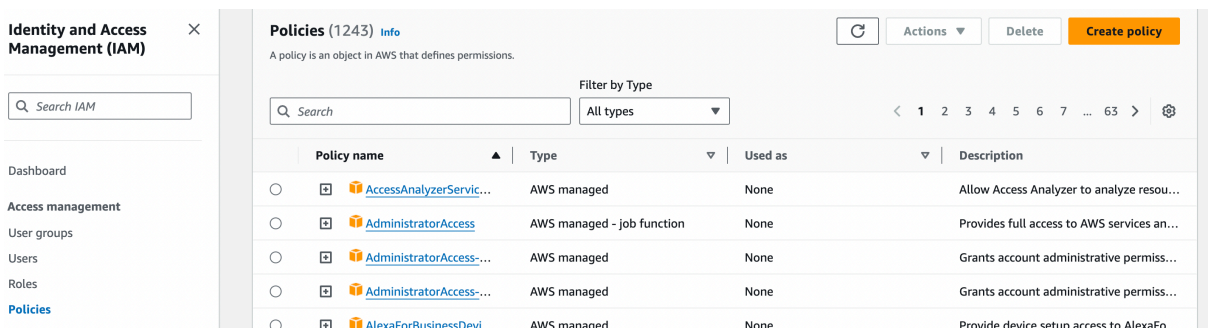
```
mahendranselvakumar@Mahendrans-MBP ~ % vault secrets list
Path          Type         Accessor              Description
----          ----         --------              -----------
aws/          aws          aws_6975c27a          n/a
cubbyhole/    cubbyhole    cubbyhole_86589c0d    per-token private 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
mahendranselvakumar@Mahendrans-MBP ~ %
```

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

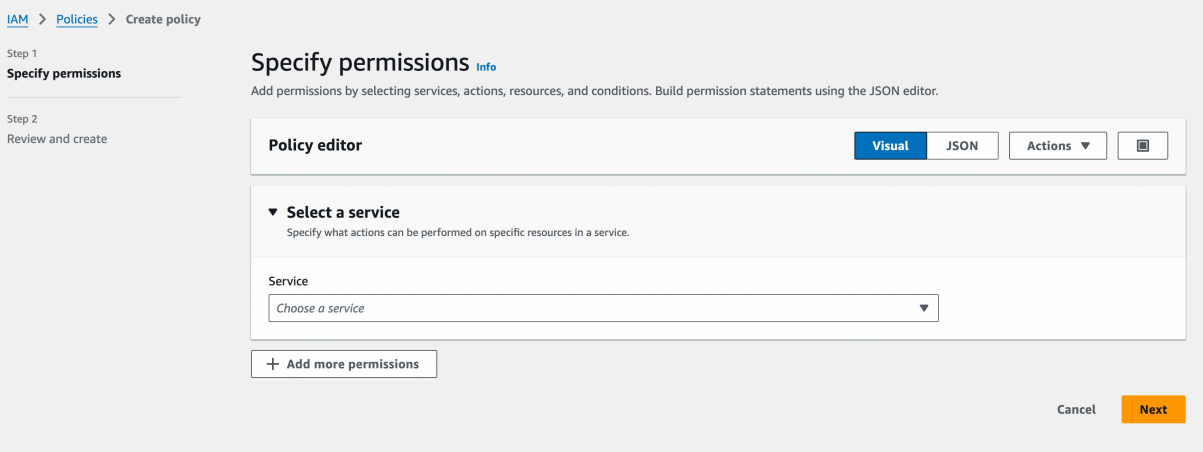
3a.Create an IAM policy to manage dynamic IAM user credentials



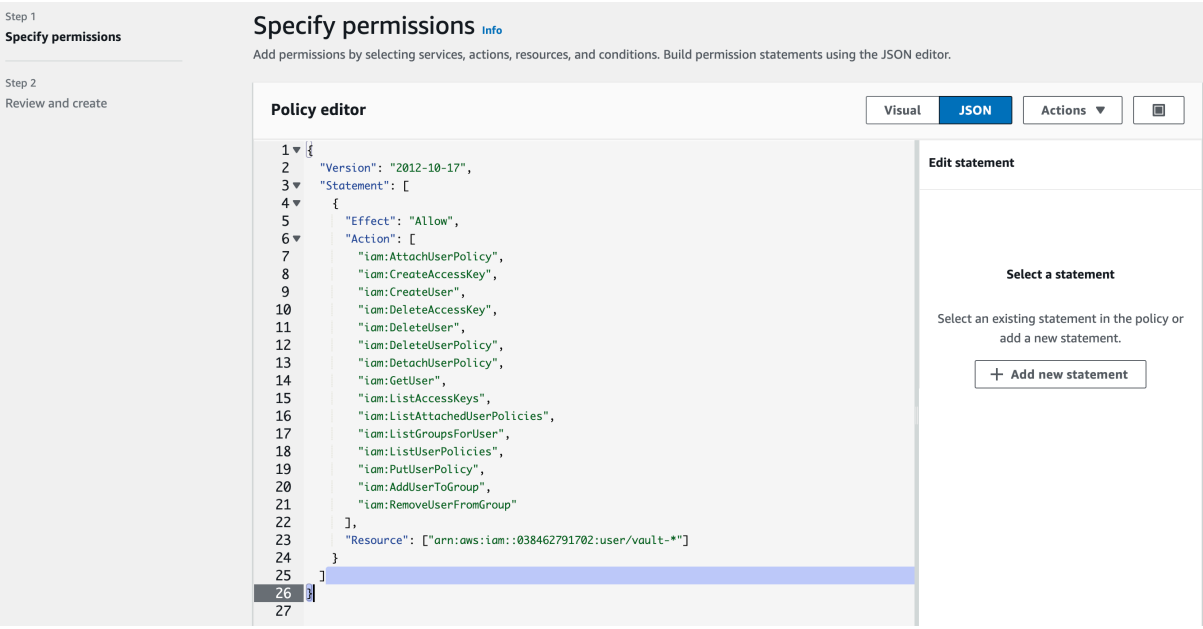
Navigate to **Policies** and click **Create Policy**



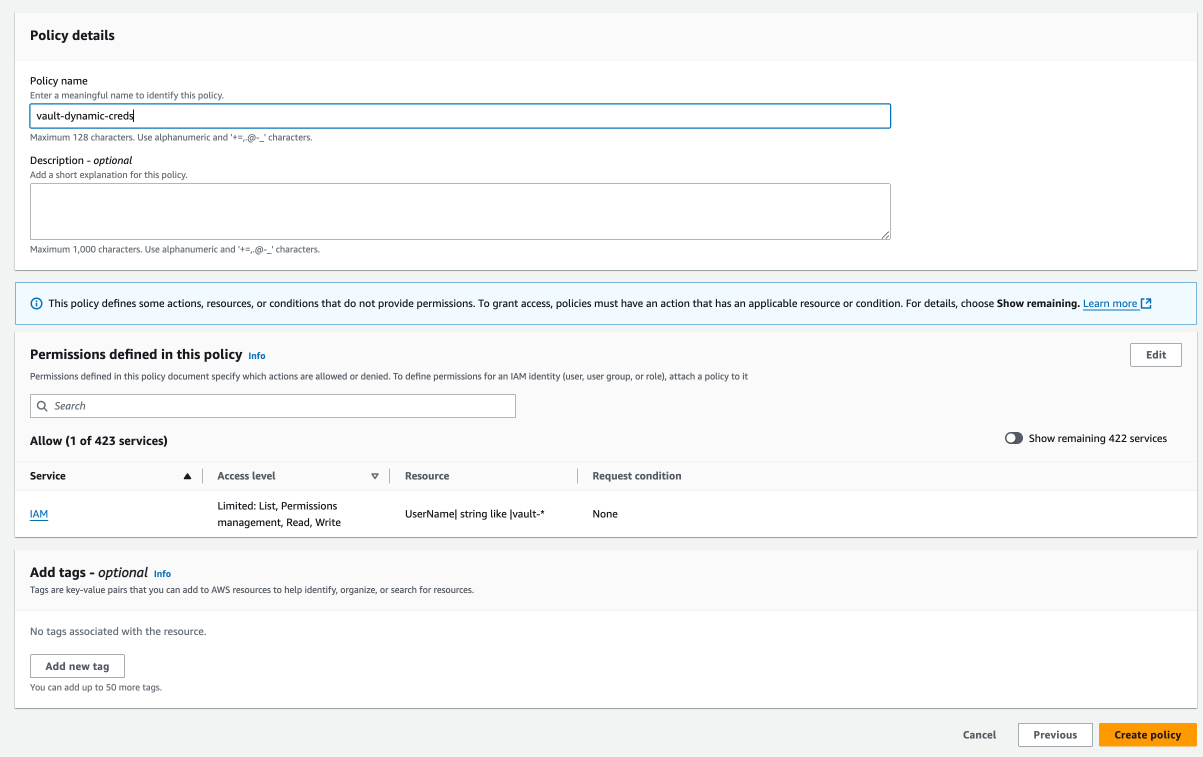
Switch the policy editor to JSON mode, and copy the IAM permission policy from the HashiCorp website (https://developer.hashicorp.com/vault/docs/secrets/aws#iam-permissions-policy-for-vault)



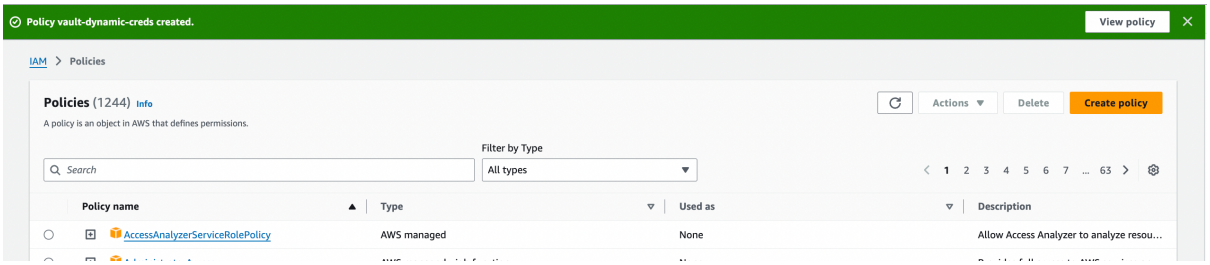
Modify the account number in the **Resource** section of the policy to match your AWS account number



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

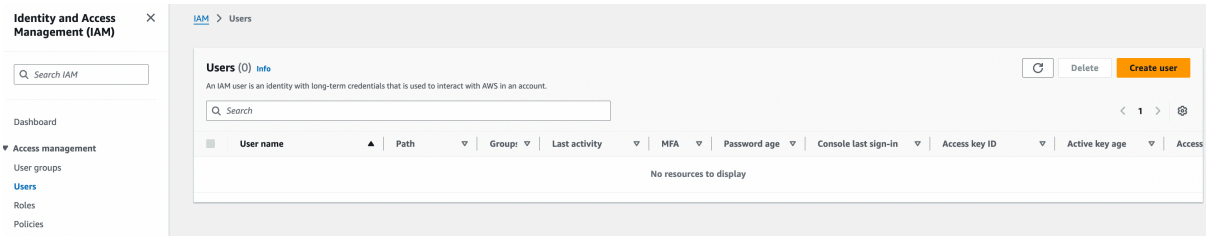


The Policy has been created

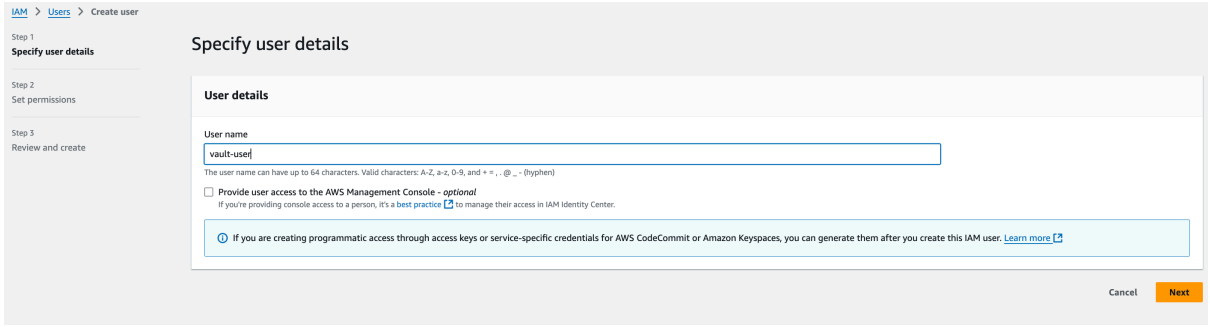


3b.Create an IAM user and assign the newly created IAM policy to the user

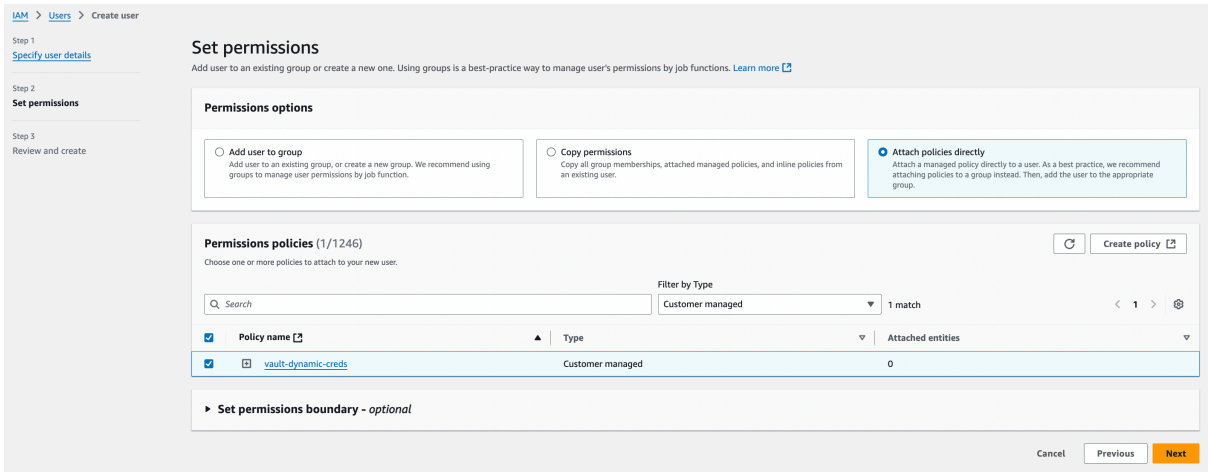
Go to **Users** in IAM and click **Create user**



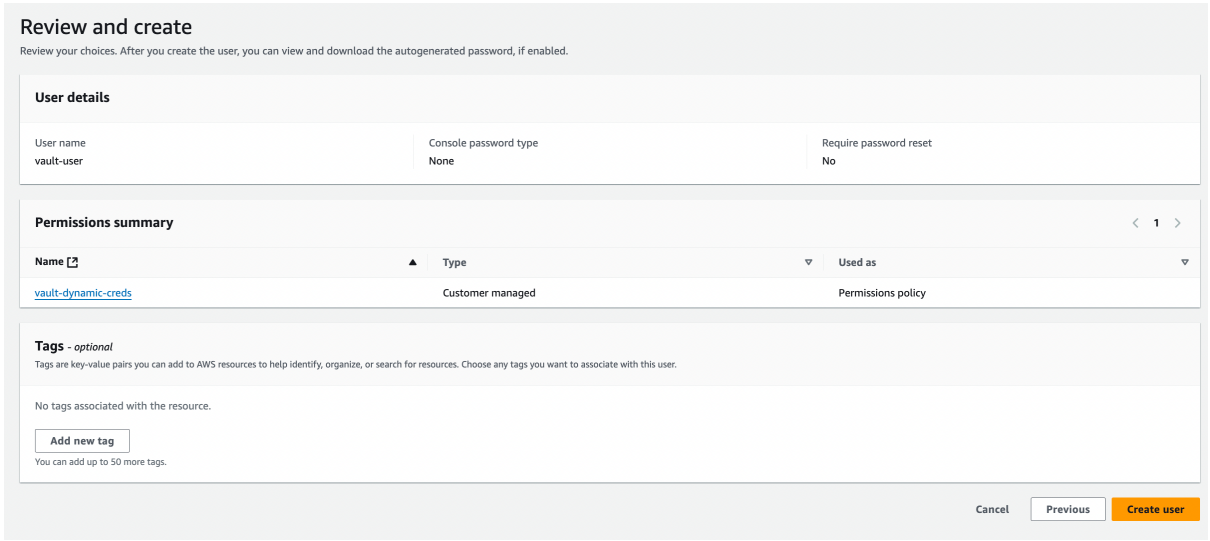
Enter the user name and click “Next”



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

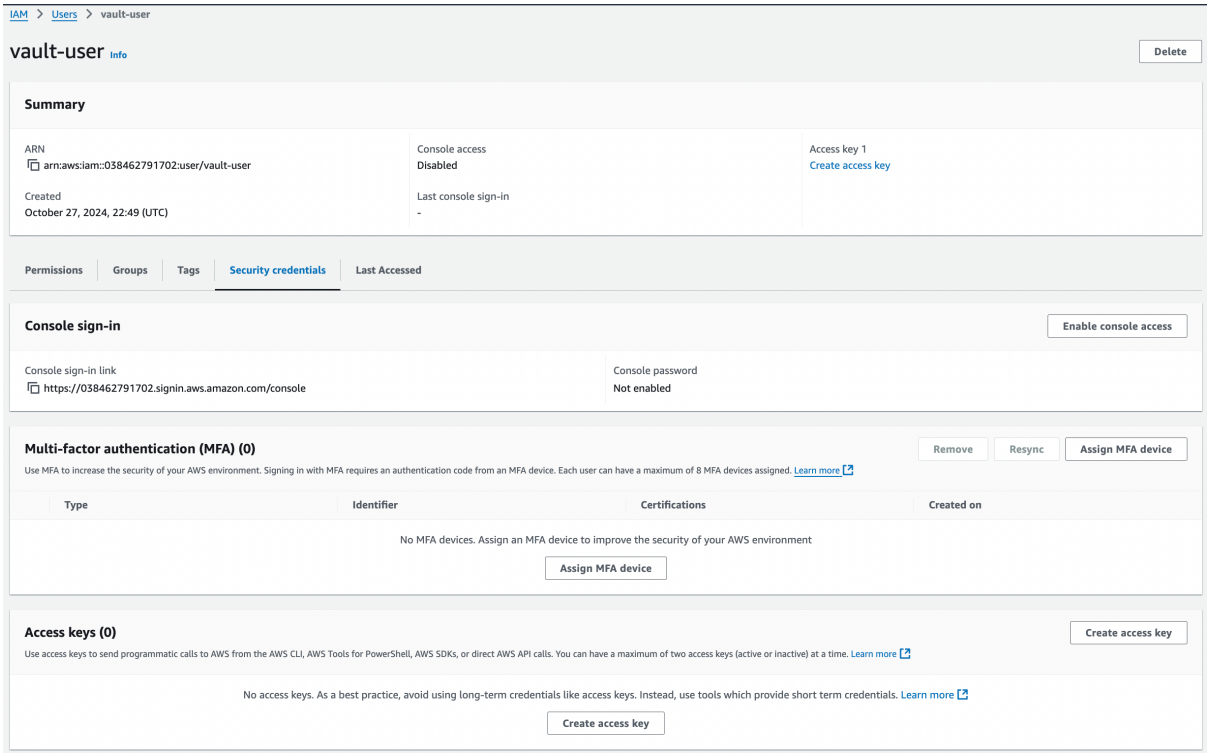


Review and click Create user

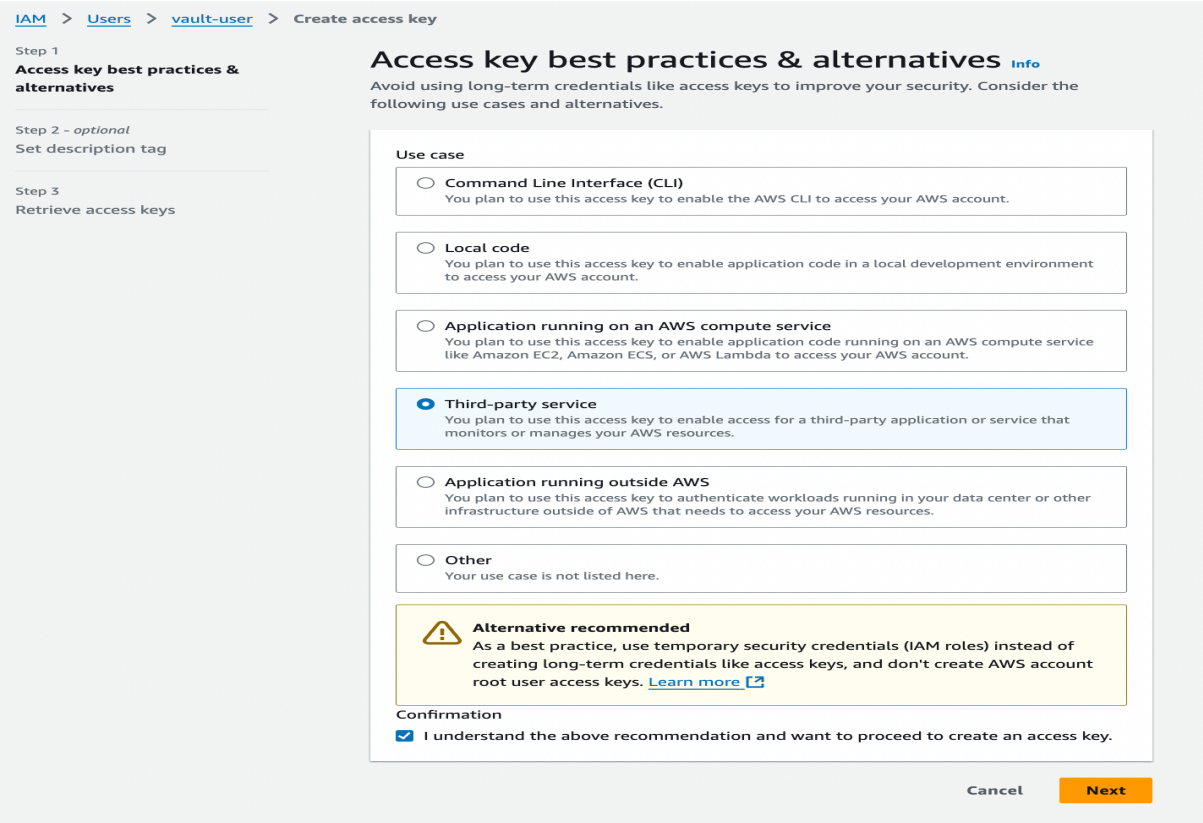


3c. Create Programmatic access to the user

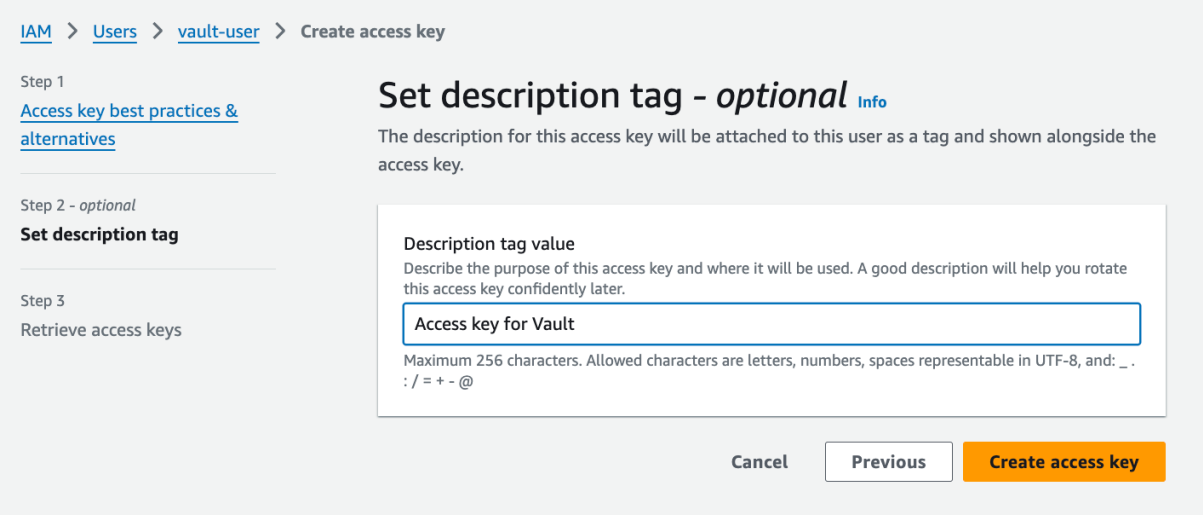
Go to the created user and click Create access key



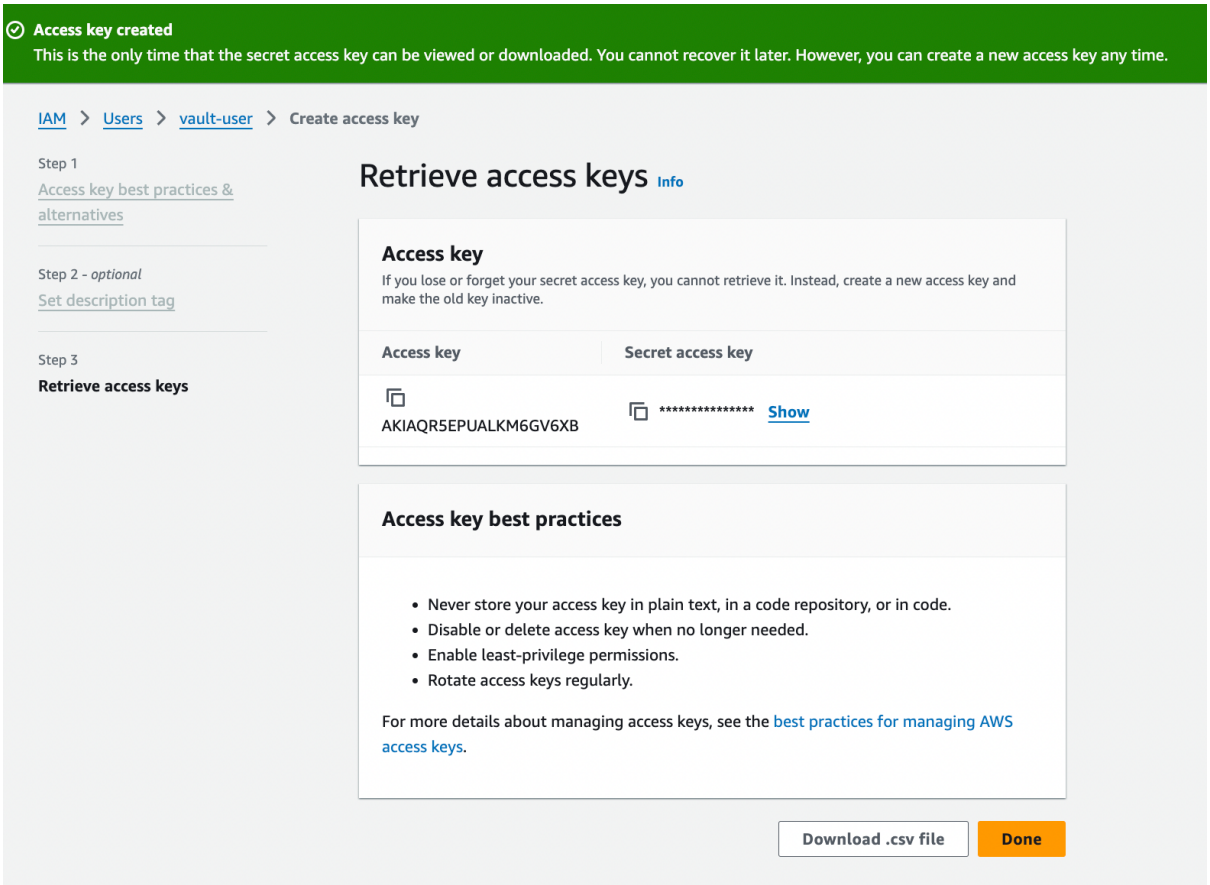
Select **Third-party service**, check the box for **I understand**, and click **Next**



Provide the **description tag value** and click **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



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

```
mahendranselvakumar@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
mahendranselvakumar@Mahendrans-MBP ~ %
```

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

```
mahendranselvakumar@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 }}
mahendranselvakumar@Mahendrans-MBP ~ %
```

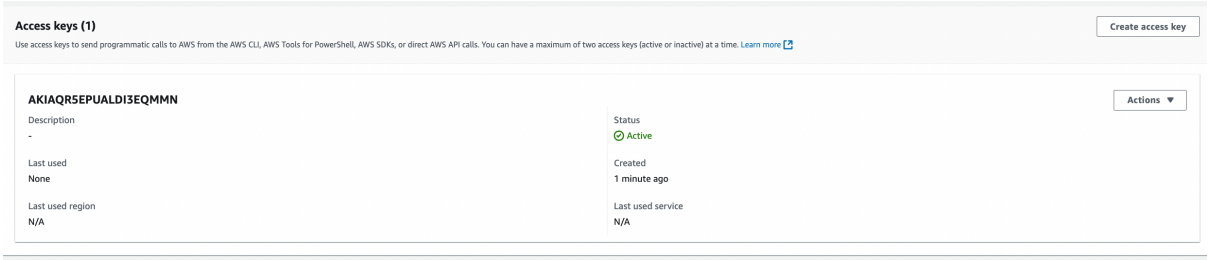
Rotate the AWS Credentials by using the **vault write -f aws/config/rotate-root**

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

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

```
mahendranselvakumar@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 }}
mahendranselvakumar@Mahendrans-MBP ~ %
```

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



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

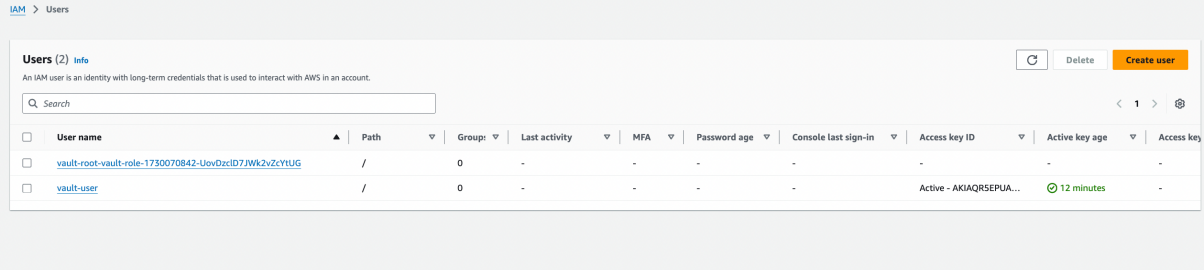
Here, I have used **AmazonEC2ReadOnlyAccess**, but you can use any role

```
mahendranselvakumar@Mahendrans-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
mahendranselvakumar@Mahendrans-MBP ~ %
```

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

```
mahendranselvakumar@Mahendrans-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         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
session_token      <nil>
mahendranselvakumar@Mahendrans-MBP ~ %
```

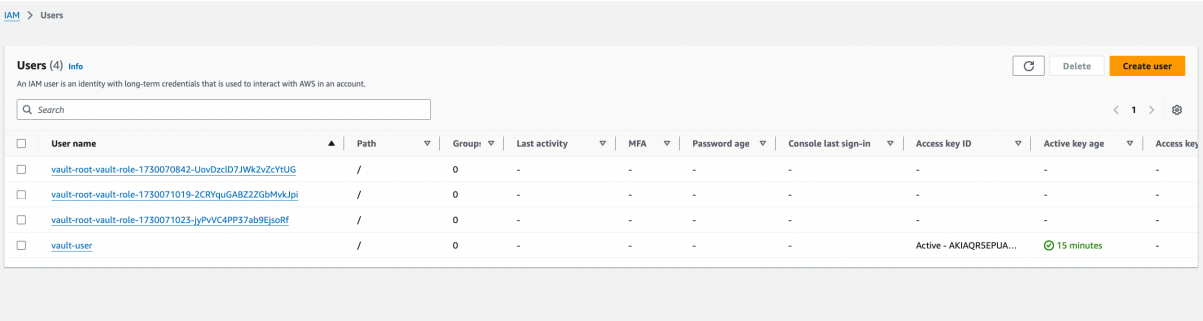
A new IAM user is created when you run the command **vault read aws/creds/vault-role**



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

```
mahendranselvakumar@Mahendrans-MBP ~ % vault read aws/creds/vault-role
Key                Value
---                -----
lease_id           aws/creds/vault-role/CVtWVVp5FpaOFD6o3kKkJ7Te
lease_duration     768h
lease_renewable    true
access_key         AKIAQR5EPUALDQHRT7WX
secret_key         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
session_token      <nil>
mahendranselvakumar@Mahendrans-MBP ~ % vault read aws/creds/vault-role
Key                Value
---                -----
lease_id           aws/creds/vault-role/Y2GapAxEvO8kh4gX5ce2WlsR
lease_duration     768h
lease_renewable    true
access_key         AKIAQR5EPUALIXKWN4CB
secret_key         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
session_token      <nil>
mahendranselvakumar@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



Revoke Vault credential:

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

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

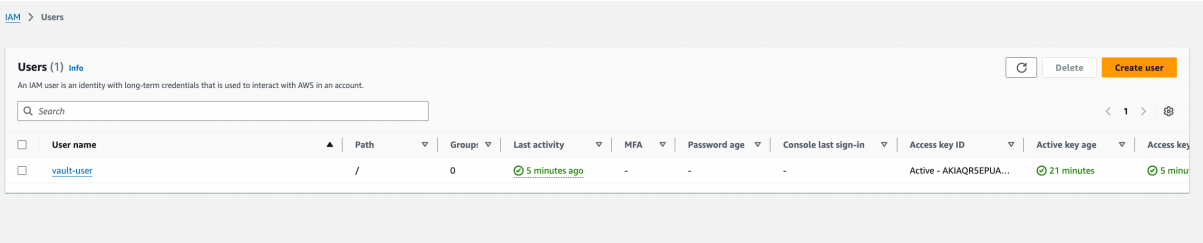
vault lease revoke: This command is used to revoke a specific lease in Vault

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**

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

Now all the users have been deleted



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

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