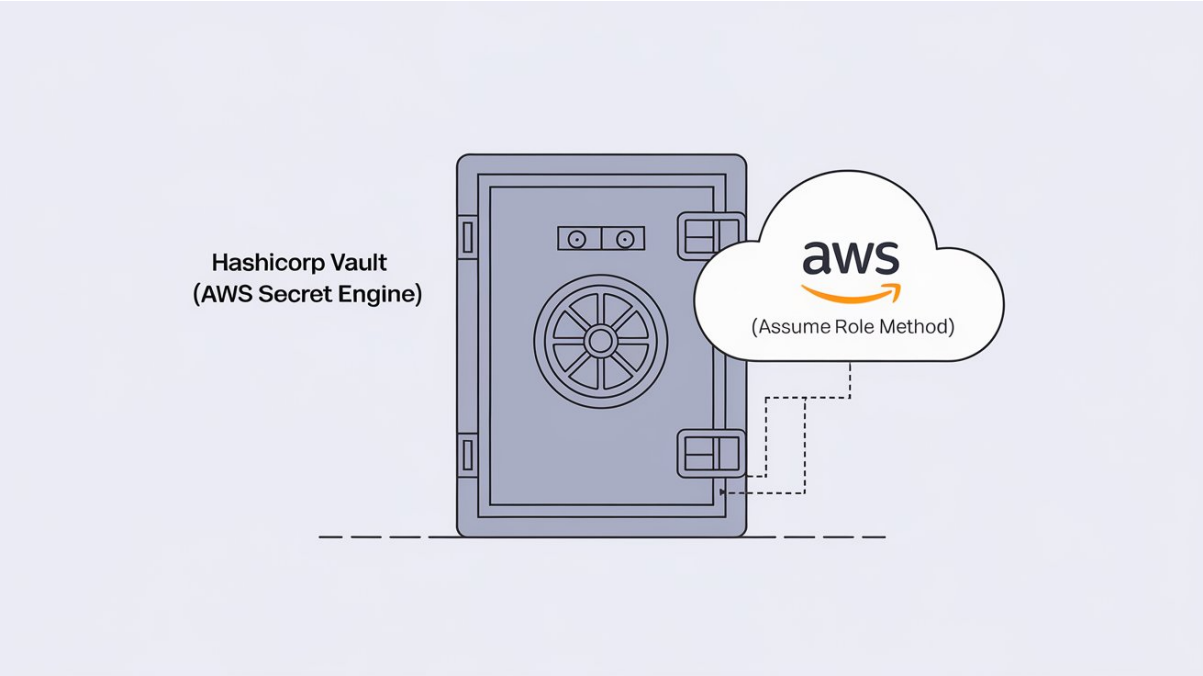
# Step-by-Step Guide to Configuring Dynamic AWS Credentials Using Assumed Role Method in HashiCorp Vault




**By**

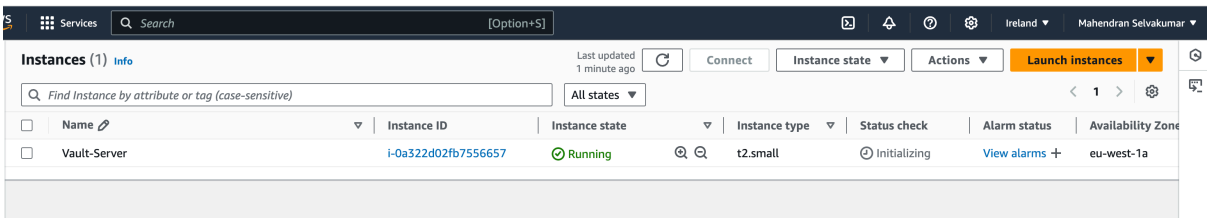
**Mahendran Selvakumar**

https://devopstronaut.com/

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

## Step1: Install Hashicorp Vault on Linux instance

Create an Amazon Linux EC2 instance for Vault




Login to Vault Server

```
mahendranselvakumar@Mahendrans-MBP Downloads % ssh -i "kprwindows.pem" ec2-user@ec2-108-129-182-13.eu-west-1.compute.amazonaws.com
The authenticity of host 'ec2-108-129-182-13.eu-west-1.compute.amazonaws.com (108.129.182.13)' can't be established.
ED25519 key fingerprint is SHA256:e2DYv01RK44oGXLDmBj5dc8Q1TmFDCJNcR2bSBzmqtw.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added 'ec2-108-129-182-13.eu-west-1.compute.amazonaws.com' (ED25519) to the list of known hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-172-31-42-27 ~]$
```

Install **yum-utils** and **shadow-utils** on the Linux Instance

- yum-utils: This is a package that provides utilities and tools for managing YUM repositories, installing, removing, and updating packages, and more.
- shadow-utils: This is a package that provides essential utilities for managing user accounts and passwords, including useradd, usermod, passwd, and others.

```
[ec2-user@ip-172-31-42-27 ~]$ sudo su -
[root@ip-172-31-42-27 ~]# sudo yum install -y yum-utils shadow-utils
Last metadata expiration check: 0:04:37 ago on Thu Nov 21 00:33:44 2024.
Package dnf-utils-4.3.0-13.amzn2023.0.4.noarch is already installed.
Package shadow-utils-2:4.9-12.amzn2023.0.4.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
[root@ip-172-31-42-27 ~]#
```

Add hashicorp repo to the Yum Package manager

```
[root@ip-172-31-42-27 ~]# sudo yum-config-manager --add-repo https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo
Adding repo from: https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo
[root@ip-172-31-42-27 ~]#
```

Install Vault using Yum package using this command **sudo yum install vault -y**


Verify the Vault installation with **vault --version** command

```
[root@ip-172-31-42-27 ~]# vault --version
Vault v1.18.1 (f479e5c85462477c9334564bc8f69531cdb03b65), built 2024-10-29T14:21:31Z
[root@ip-172-31-42-27 ~]# vault
Usage: vault <command> [args]

Common commands:
    read            Read data and retrieves secrets
    write           Write data, configuration, and secrets
    delete          Delete secrets and configuration
    list            List data or secrets
    login           Authenticate locally
    agent           Start a Vault agent
    server          Start a Vault server
    status          Print seal and HA status
    unwrap          Unwrap a wrapped secret

Other commands:
    audit                     Interact with audit devices
    auth                      Interact with auth methods
    debug                     Runs the debug command
    events
    hcp
    kv                        Interact with Vault's Key-Value storage
    lease                     Interact with leases
    monitor                   Stream log messages from a Vault server
    namespace                 Interact with namespaces
    operator                  Perform operator-specific tasks
    patch                     Patch data, configuration, and secrets
    path-help                 Retrieve API help for paths
    pki                       Interact with Vault's PKI Secrets Engine
    plugin                    Interact with Vault plugins and catalog
    policy                    Interact with policies
    print                     Prints runtime configurations
    proxy                     Start a Vault Proxy
    secrets                   Interact with secrets engines
    ssh                       Initiate an SSH session
    token                     Interact with tokens
    transform                 Interact with Vault's Transform Secrets Engine
    transit                   Interact with Vault's Transit Secrets Engine
    version-history           Prints the version history of the target Vault server
[root@ip-172-31-42-27 ~]#
```

## Step 2: Start Vault with Dev Server Mode

We can run vault instance as dev mode using **vault server -*dev*** command

```
[root@ip-172-31-42-27 ~]# 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: BASH_FUNC_which%%, HISTCONTROL, HISTSIZE, HOME, HOSTNAME, LANG, LESSOPEN, LOGNAME, LS_COLORS, MAIL, PATH, PWD, SHELL, SHLVL, SYSTEMD_COLORS,
 S_COLORS, TERM, USER, _, which_declare
              Go Version: go1.22.8
              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: true, enabled: false
           Recovery Mode: false
                 Storage: inmem
                 Version: Vault v1.18.1, built 2024-10-29T14:21:31Z
             Version Sha: f479e5c85462477c9334564bc8f69531cdb03b65

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

```
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: at/ytj5ESqD4wBSuJ7EIIKvgl6Tpv31U8KqewN0EwnI=
Root Token: hvs.P1O6ZCfFnhNW453CxuGcbhbl

Development mode should NOT be used in production installations!
```

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.

Log in to Vault using another terminal because it is in dev mode

```
mahendranselvakumar@Mahendrans-MBP Downloads % ssh -i "kprwindows.pem" ec2-user@ec2-108-129-182-13.eu-west-1.compute.amazonaws.com
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Thu Nov 21 00:35:00 2024 from 176.248.232.84
[ec2-user@ip-172-31-42-27 ~]$ sudo su -
Last login: Thu Nov 21 00:37:52 UTC 2024 on pts/1
[root@ip-172-31-42-27 ~]#
```

**Set Environment Variable**

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

```
[root@ip-172-31-42-27 ~]# export VAULT_ADDR='http://127.0.0.1:8200'
[root@ip-172-31-42-27 ~]#
```

Run the **vault status** command, and you will see that the Vault server's seal type is Shamir. It is initialized and automatically unsealed. The storage type is in-memory, which means it is not suitable for production use

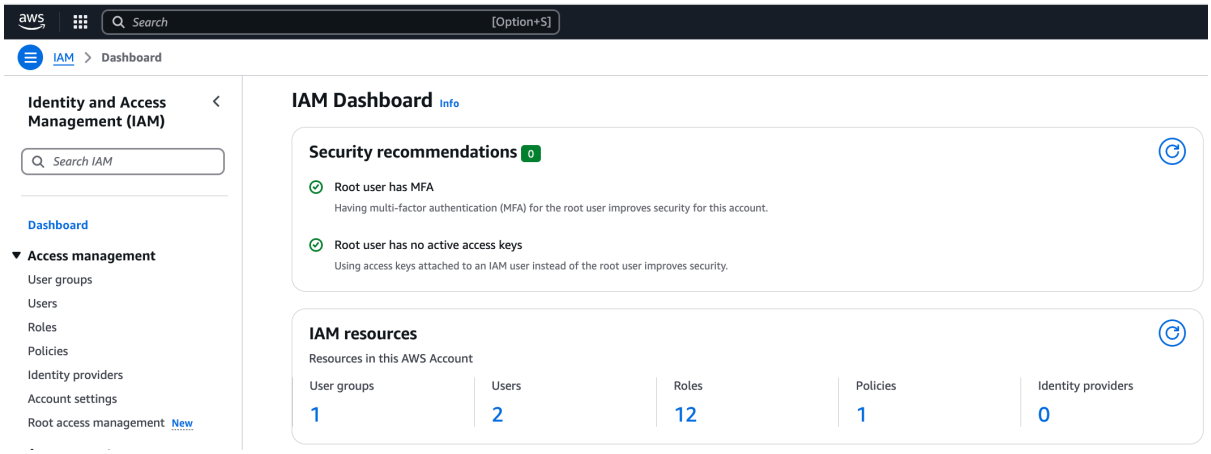
```
[root@ip-172-31-42-27 ~]# vault status
Key             Value
---             -----
Seal Type       shamir
Initialized     true
Sealed          false
Total Shares    1
Threshold       1
Version         1.18.1
Build Date      2024-10-29T14:21:31Z
Storage Type    inmem
Cluster Name    vault-cluster-04b70bbd
Cluster ID      c4acbf7c-08c4-a6de-c79f-f49aa2e033ae
HA Enabled      false
[root@ip-172-31-42-27 ~]#
```

Run the vault secrets list command to view the list of enabled secrets engines

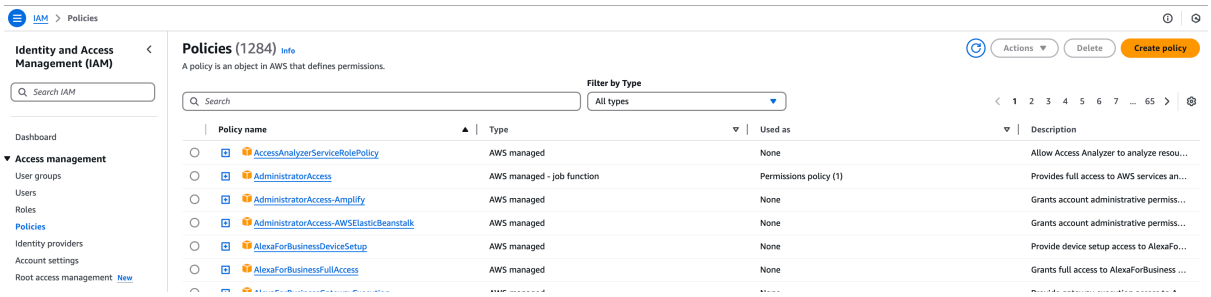
```
[root@ip-172-31-42-27 ~]# vault secrets list
Path          Type         Accessor              Description
----          ----         --------              -----------
cubbyhole/    cubbyhole    cubbyhole_5794adb0    per-token private secret storage
identity/     identity     identity_a09ee25f     identity store
secret/       kv           kv_1887f107           key/value secret storage
sys/          system       system_4f9b7cc1       system endpoints used for control, policy and debugging
[root@ip-172-31-42-27 ~]#
```

## Step 3: Create an IAM Policy and Role in the Vault Server Account

Go to the IAM Dashboard and click on **Policies**




Click on **Create policy**

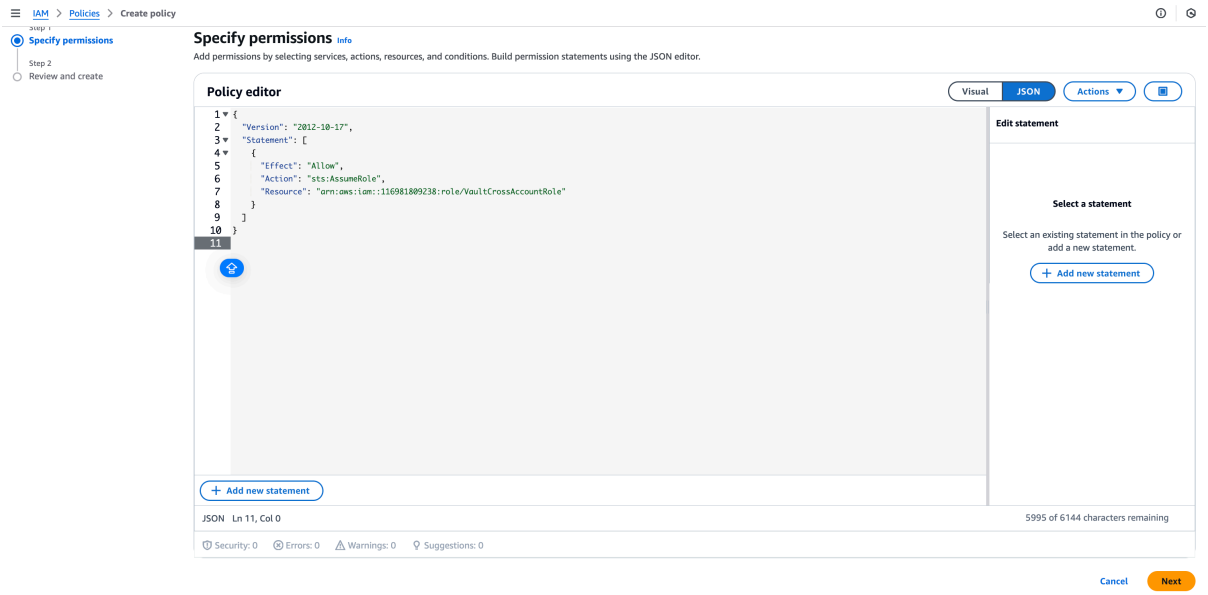



Create a code for an IAM policy that includes the target account ID and role

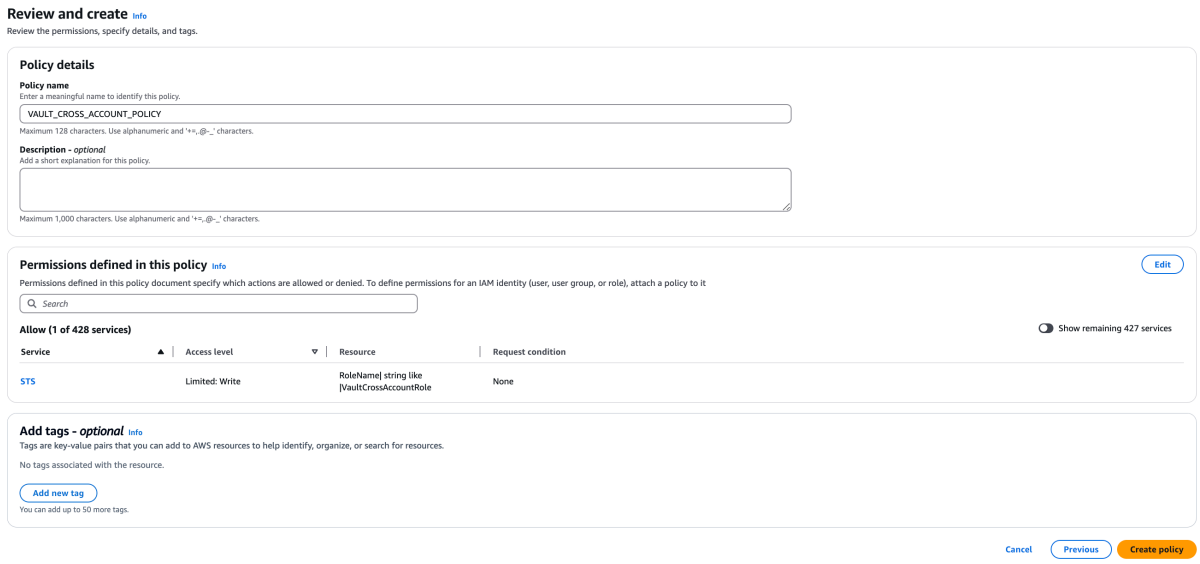
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Resource": "arn:aws:iam::116981809238:role/VaultCrossAccountRole"
    }
  ]
}
```

- Replace the account number with the AWS Account ID of the target account
- Replace the role name with the name of the role in the target account

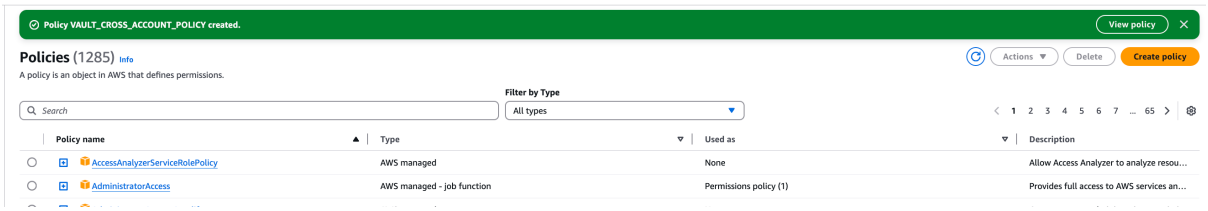
Switch the Policy editor to **JSON**, paste the above code, and click **Next**



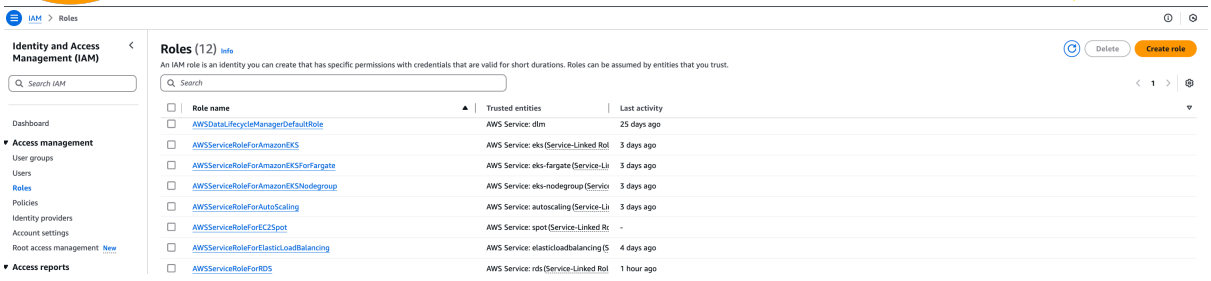

Review the policy details and click **Create policy**



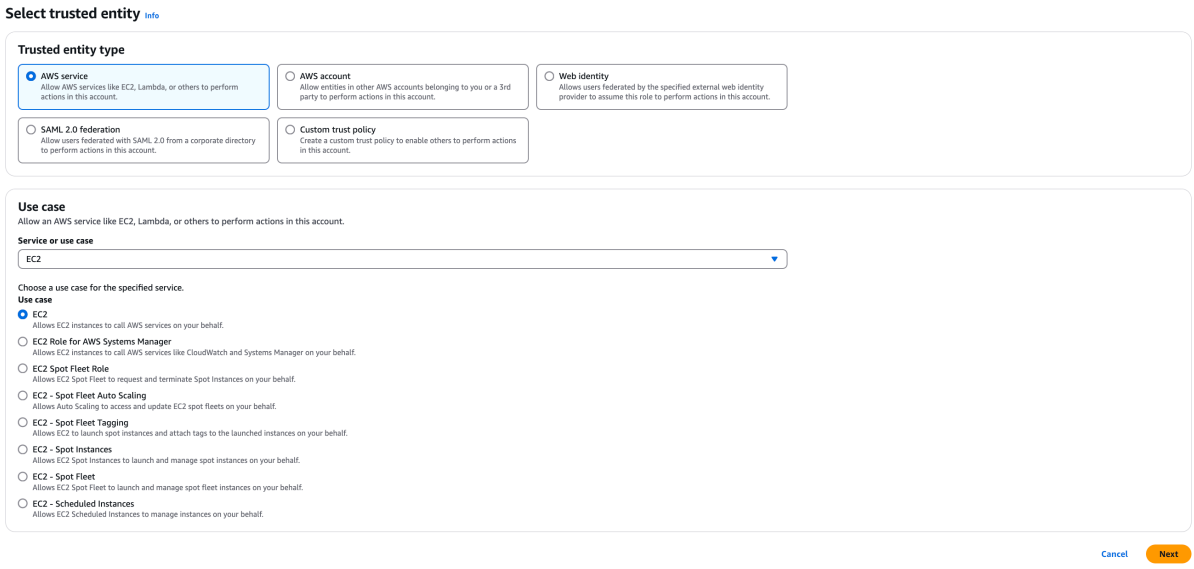

Now Policy has been created



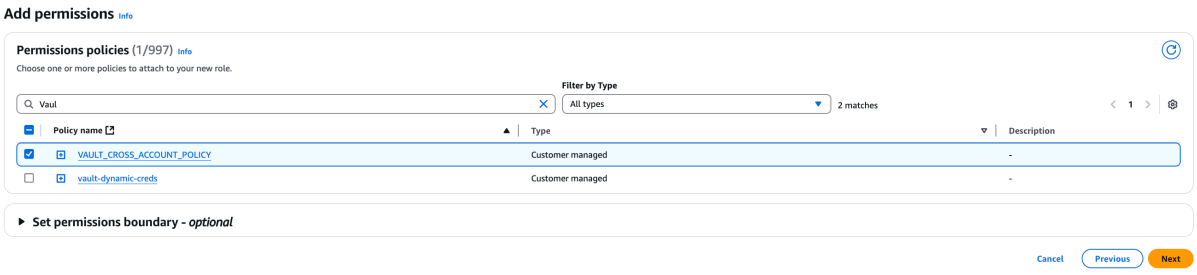

Go to **Roles** and click on **Create role**




## Choose **AWS Service** as the trusted entity type, select **EC2**, and then click **Next**



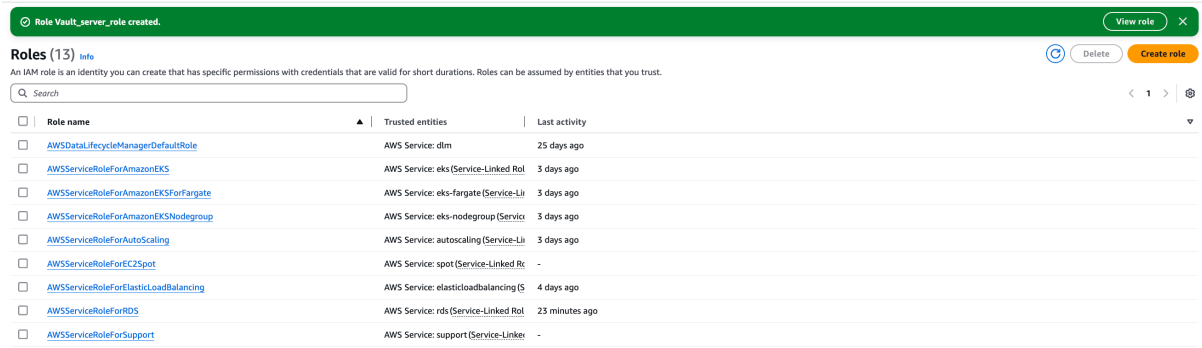

## Select the policy created for Vault cross-account access and click **Next**




## Enter the role name and click **Create role**


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

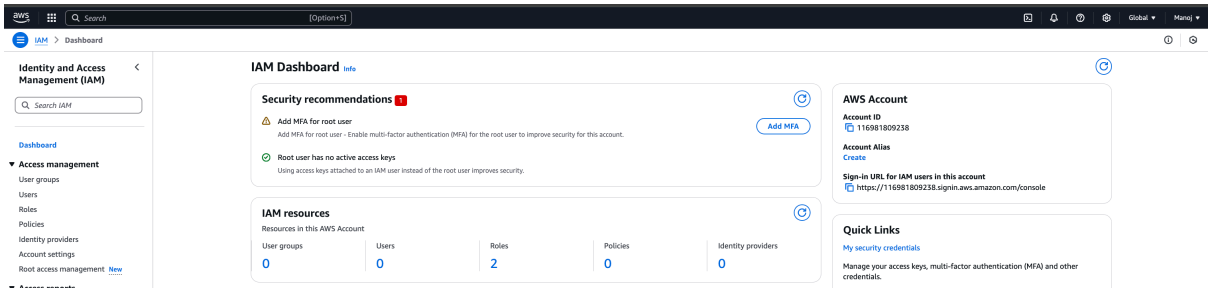
The IAM role has now been successfully created.



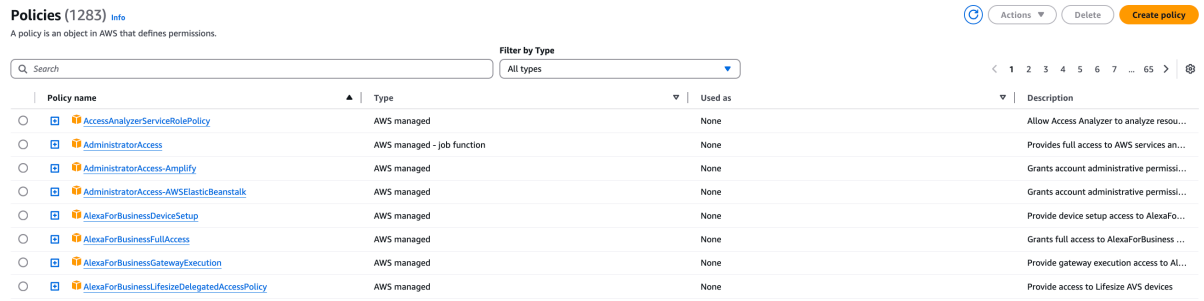

Attach this IAM role to the Vault instance

## Step 4: Create an IAM Policy and Role in the Client/Another Account

Go to the other AWS account, navigate to **IAM**, and click on **Policies**

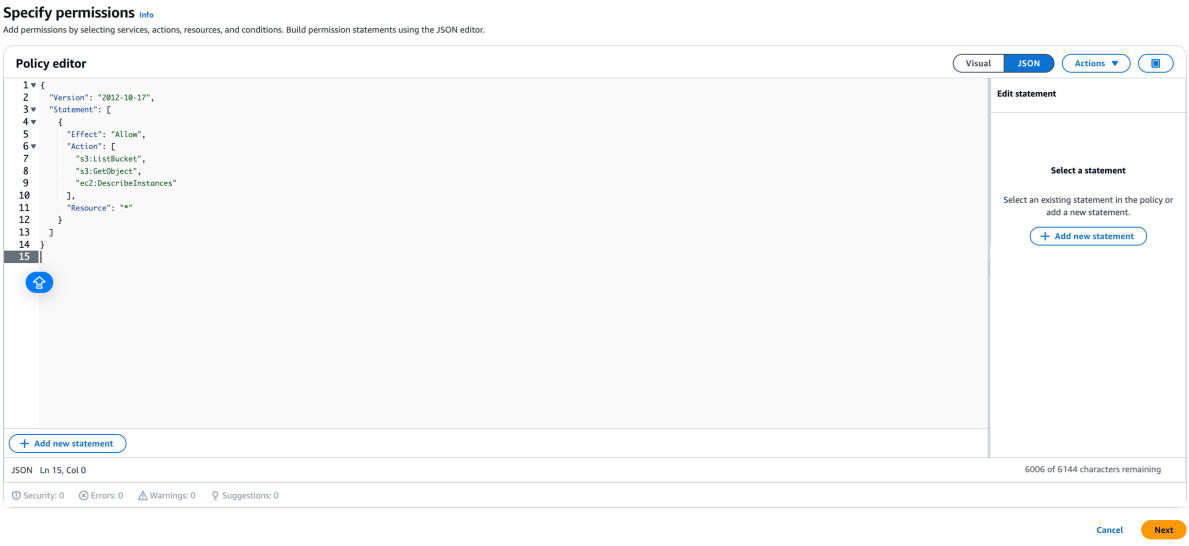



Click on **Create policy**

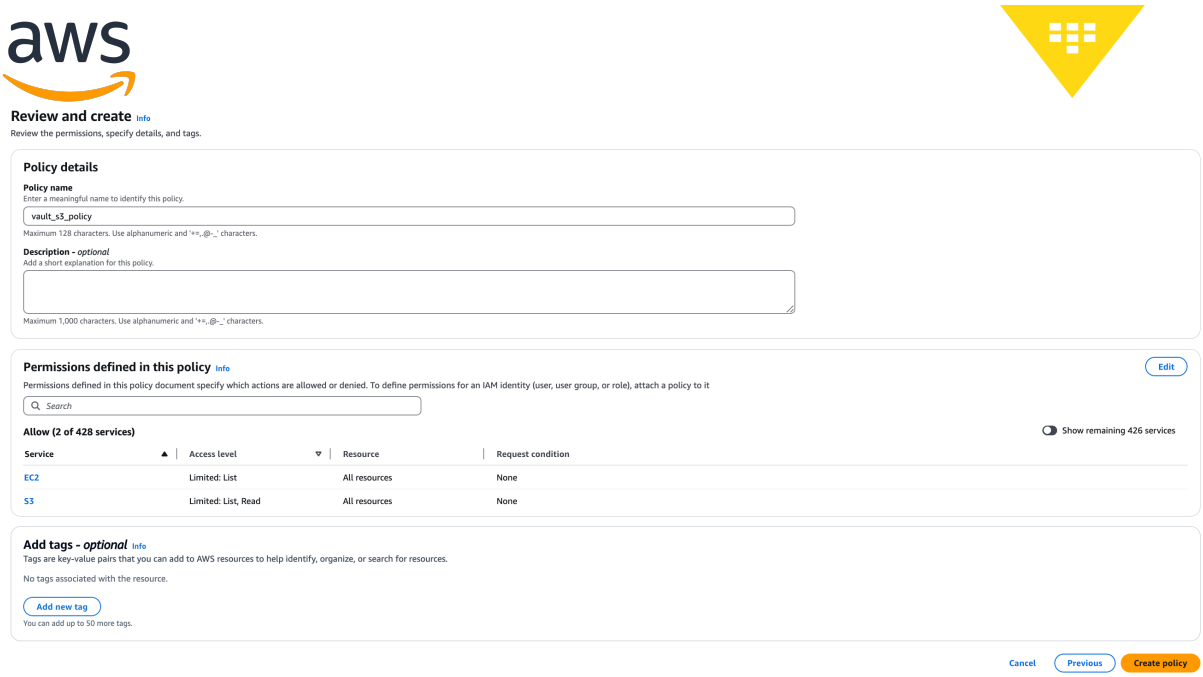



Create code for an IAM policy with permissions for S3 bucket and EC2 instance access. Currently, it uses *, but in a production environment, restrict the policy to specific resources and actions

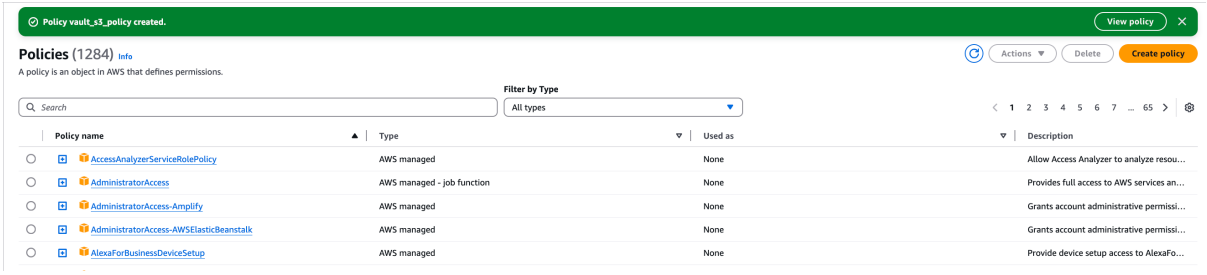

Switch the policy editor to **JSON**, paste the modified code, and click **Next**



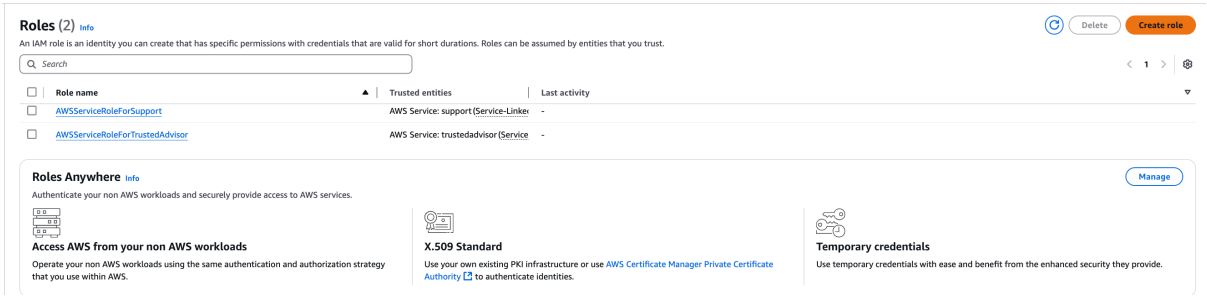

Enter the policy name and click **Create policy**



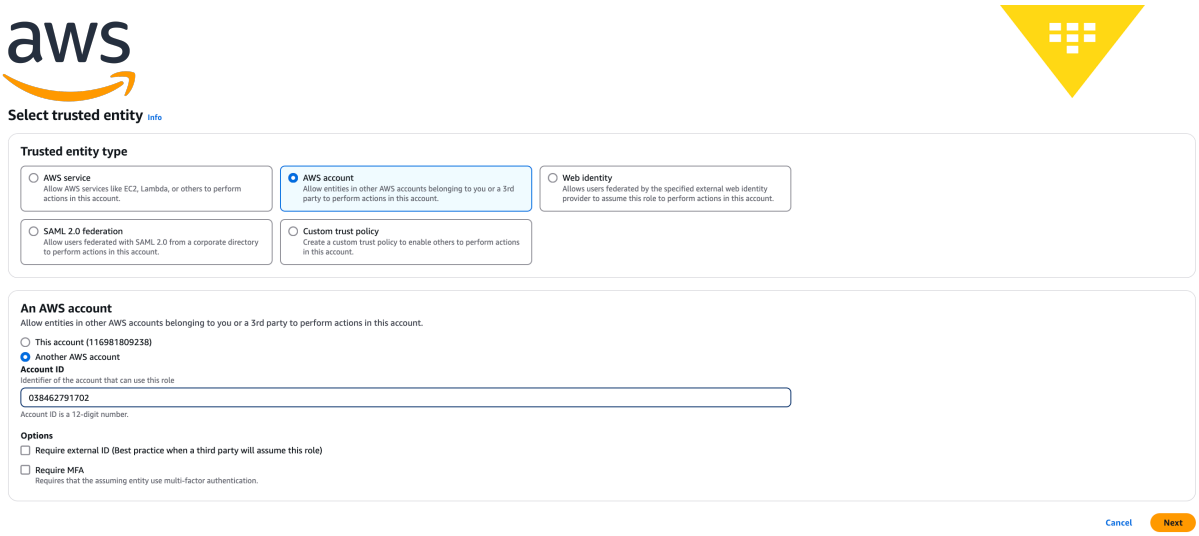

The policy has been successfully created with S3 and EC2 access



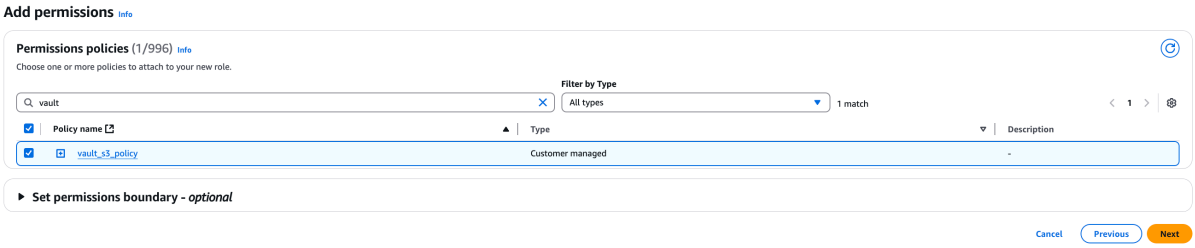

Go to **Roles** and click on **Create role**



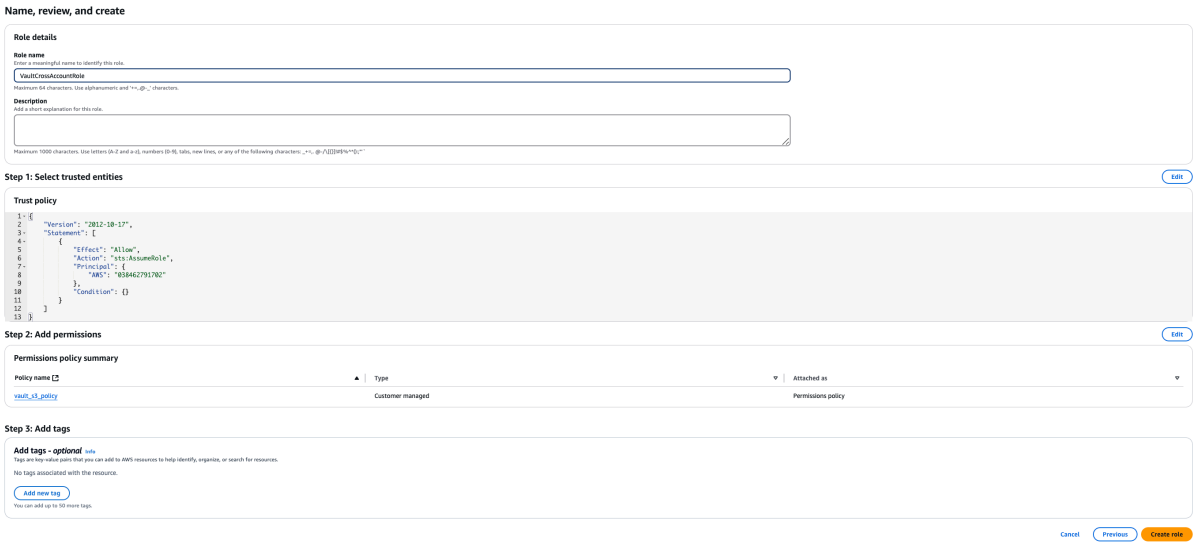

Select **Trusted Entity type** as **AWS Account**, choose **Another AWS account**, enter the **Account ID**, and click **Next**



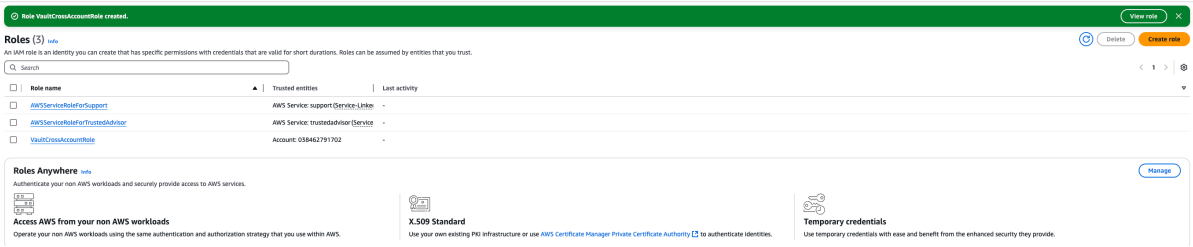

In the **Permissions** section, select the policy you just created and click **Next**




Enter the role name and click **Create role**




The role has been successfully created




## Step 5:Define Vault Role for Dynamic Secrets

Log in to Vault using the root token provided during dev mode initialization with this command **vault login <root-token>**

```
[root@ip-172-31-42-27 ~]# vault login hvs.P1O6ZCfFnhNW453CxuGcbhbl
Success! You are now authenticated. The token information displayed below
is already stored in the token helper. You do NOT need to run "vault login"
again. Future Vault requests will automatically use this token.

Key                  Value
---                  -----
token                hvs.P1O6ZCfFnhNW453CxuGcbhbl
token_accessor       fHBbhWLQ4C9WEOLW4BcaA623
token_duration       ∞
token_renewable      false
token_policies       ["root"]
identity_policies    []
policies             ["root"]
[root@ip-172-31-42-27 ~]#
```

### Enable AWS Secrets Engine using the CLI

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

```
[root@ip-172-31-42-27 ~]# vault secrets enable aws
Success! Enabled the aws secrets engine at: aws/
[root@ip-172-31-42-27 ~]#
```

### Configure AWS Credentials in vault for accessing AWS Services

Vault needs IAM credentials from the source account to operate. These credentials should have sts:AssumeRole permissions to access the target account

```
[root@ip-172-31-42-27 ~]# vault write aws/config/root \
access_key=AKIAQR5EPUALKGQDCHXD \
secret_key=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 \
region=eu-west-1
Success! Data written to: aws/config/root
[root@ip-172-31-42-27 ~]#
```

Run this command **vault write aws/roles/s3_bucket_access role_arns=arn:aws:iam::116981809238:role/VaultCrossAccountRole** to define a role in Vault that will dynamically assume the role in the target account

```
[root@ip-172-31-42-27 ~]# vault write aws/roles/s3_bucket_access role_arns=arn:aws:iam::116981809238:role/VaultCrossAccountRole credential_type=assumed_role
Success! Data written to: aws/roles/s3_bucket_access
[root@ip-172-31-42-27 ~]#
```

Run the **vault write aws/sts/s3_bucket_access -ttl=60m** command to generate AWS credentials with a time-to-live (TTL) of 60 minutes

```
[root@ip-172-31-42-27 ~]# vault write aws/sts/s3_bucket_access -ttl=60m
Command flags must be provided before positional arguments. The following arguments will not be parsed as flags: [-ttl=60m]
WARNING! The following warnings were returned from Vault:

  * Endpoint ignored these unrecognized parameters: [-ttl]

Key                Value
---                -----
lease_id           aws/sts/s3_bucket_access/uOvX2j7TDnVjoCESdl2fcZa4
lease_duration     1h
lease_renewable    false
access_key         ASIARWPFIYBLETFUY7JD
arn                arn:aws:sts::116981809238:assumed-role/VaultCrossAccountRole/vault-root-s3_bucket_access-1732154239-7j3DLjVfMHmbM9aGUu4u
secret_key         3aWZumkSFs95hqbATnGtID5Hk9e/XEpNFo1fdTm7
security_token     FwoGZXIvYXdzEBMaDFVVh8NUbr9Hf/8mOiLfAasIBnDtOKwZQvQt1MOjXCaQxnPZmAUhRvtXMMOdXh5KFin2EFntBdisJENdLBEB0iIWtGE2dQDEOBQ4wqXjhY1QSiPJDz6tkLjM4k7rUgb8Yhd
MzrnGAx7KGo5XZb0H7wFNGsFWzdhqBbCsDfeJANqRYGxU0eMTH5vfJLjxBDE4pCkoCeZntMQ1OakV8Hk1E6pTs5618GzvnvDXhmAaaAyuuJJBqs3uGVevQEFNfYHNckJf5NzQFntEAvGILltApf+Ork68xT3FPA6Jl8zxH
ShslfzbYTZBmSn3y6wr4mMo/6b6uQYyLXo45DSA1IU8gwGLH9kz6VVG9u/KAEDykLAPohjNIRvhWuKChtngxfvy6fWnEg==
session_token      FwoGZXIvYXdzEBMaDFVVh8NUbr9Hf/8mOiLfAasIBnDtOKwZQvQt1MOjXCaQxnPZmAUhRvtXMMOdXh5KFin2EFntBdisJENdLBEB0iIWtGE2dQDEOBQ4wqXjhY1QSiPJDz6tkLjM4k7rUgb8Yhd
MzrnGAx7KGo5XZb0H7wFNGsFWzdhqBbCsDfeJANqRYGxU0eMTH5vfJLjxBDE4pCkoCeZntMQ1OakV8Hk1E6pTs5618GzvnvDXhmAaaAyuuJJBqs3uGVevQEFNfYHNckJf5NzQFntEAvGILltApf+Ork68xT3FPA6Jl8zxH
ShslfzbYTZBmSn3y6wr4mMo/6b6uQYyLXo45DSA1IU8gwGLH9kz6VVG9u/KAEDykLAPohjNIRvhWuKChtngxfvy6fWnEg==
ttl                59m59s
[root@ip-172-31-42-27 ~]#
```

This will output temporary credentials (access_key, secret_key, and session_token) that can be used to access the S3 bucket

**Step 6:Verify the Temporary Credentials**

To test the temporary credentials using the AWS CLI, export them as environment variables in your local terminal

**export AWS_ACCESS_KEY_ID=ASIARWPFIYBLETFUY7JD**

**export AWS_SECRET_ACCESS_KEY=3aWZumkSFs95hqbATnGtID5Hk9e/XEpNFo1fdTm7**

**export AWS_SESSION_TOKEN=FwoGZXIvYXdzEBMaDFVVh8NUbr9Hf/8mOiLfAasIBnDtOK**



**wZQvQt1MOjXCaQxnPZmAUhRvtXMMOdXh5KFin2EFntBdisJENdLBEB0iIWtGE2dQ DEOBQ4wqXjhY1QSiPJDz6tkLjM4k7rUgb8YhdMzrnGAx7KGo5XZb0H7wFNGsFWzdh qBbCsDfeJANqRYGxU0eMTH5vfJLjxBDE4pCkoCeZntMQ1OakV8Hk1E6pTs5618Gzvn vDXhmAaaAyuuJJBqs3uGVevQEFNfYHNckJf5NzQFntEAvGILltApf+Ork68xT3FPA6Jl8 zxHShslfzbYTZBmSn3y6wr4mMo/6b6uQYyLXo45DSA1IU8gwGLH9kz 6VVG9u/KAEDykLAPohjNIRvhWuKChtngxfvy6fWnEg==**

```
mahendranselvakumar@Mahendrans-MBP ~ % export AWS_ACCESS_KEY_ID=ASIARWPFIYBLETFUY7JD
mahendranselvakumar@Mahendrans-MBP ~ % export AWS_SECRET_ACCESS_KEY=3aWZumkSFs95hqbATnGtIDSHk9e/XEpNFo1fdTm7
mahendranselvakumar@Mahendrans-MBP ~ % export AWS_SESSION_TOKEN=FwoGZXIvYXdzEBMaDFVVh8NUbr9Hf/8mOiLfAasIBnDtOKwZQvQt1MOjXCaQxnPZmAUhRvtXMMOdXh5KFin2EFntBdisJENdLBEB0iIWtGE2dQDEOBQ4wqXjhY1QSiPJDz6tkLjM4k7rUgb8YhdMzrnGAx7KGo5XZb0H7wFNGsFWzdhqBbCsDfeJANqRYGxU0eMTH5vfJLjxBDE4pCkoCeZntMQ1OakV8Hk1E6pTs5618GzvnvDXhmAaaAyuuJJBqs3uGVevQEFNfYHNckJf5NzQFntEAvGILltApf+Ork68xT3FPA6Jl8zxHShslfzbYTZBmSn3y6wr4mMo/6b6uQYyLXo45DSA1IU8gwGLH9kz6VVG9u/KAEDykLAPohjNIRvhWuKChtngxfvy6fWnEg==
mahendranselvakumar@Mahendrans-MBP ~ % aws sts get-caller-identity
{
    "UserId": "AROARWPFIYBLIM6WXF4CN:vault-root-s3_bucket_access-1732154239-7j3DLjVfMHmbM9aGUu4u",
    "Account": "116981809238",
    "Arn": "arn:aws:sts::116981809238:assumed-role/VaultCrossAccountRole/vault-root-s3_bucket_access-1732154239-7j3DLjVfMHmbM9aGUu4u"
}
mahendranselvakumar@Mahendrans-MBP ~ %
```

Run **aws s3 ls** command to get the S3 bucket lists

```
mahendranselvakumar@Mahendrans-MBP ~ % aws s3 ls
2024-11-21 02:07:27 s3-bucket-116981809238
2024-11-21 02:08:18 vault-test-s3-bucket
mahendranselvakumar@Mahendrans-MBP ~ %
```

**Conclusion:**

we have successfully set up and tested the integration of Vault with AWS for dynamic role-based access. The process involved creating and configuring IAM policies and roles across accounts, defining a role in Vault, and testing temporary credentials with the AWS CLI. This setup demonstrates how Vault can securely manage dynamic credentials for AWS resource

Keep Learning, Keep Vault-ing!!

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/