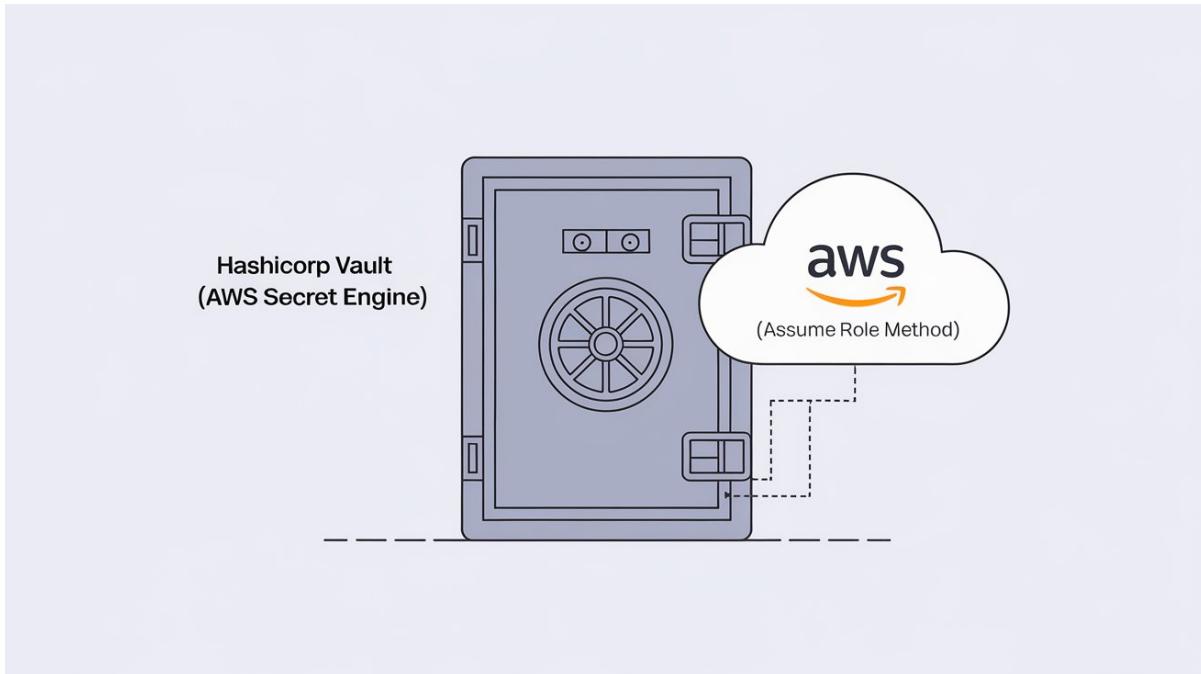




Step-by-Step Guide to Configuring Dynamic AWS Credentials Using Assumed Role Method in HashiCorp Vault



By
Mahendran Selvakumar

<https://devopstronaut.com/>



Step1: Install Hashicorp Vault on Linux instance

Create an Amazon Linux EC2 instance for Vault

The screenshot shows the AWS CloudWatch Metrics interface. A single metric named 'Vault-Server' is displayed with a value of 1. The metric has a unit of 'Count' and a timestamp of '1 minute ago'. The chart shows a single data point at time 0.

Login to Vault Server

```
mahendralselvakumar@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-' '-->
~~--. / \
~~--. / \
~~--. / \
~~--. / \
[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.

```
_/m/
[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**

```
[root@ip-172-31-42-27 ~]# sudo yum -y install vault
[download] https://releases.hashicorp.com/vault/1.18.1/vault_1.18.1-x86_64.rpm
Last metadata expiration check: 0:00:01 ago on Thu Nov 21 00:39:37 2024.
Dependencies resolved.
=====
| Package           | Architecture | Version   | Repository | Size |
|:-----|:-----|:-----|:-----|:-----|
| Installing:      |             |           |           |       |
| vault            | x86_64      | 1.18.1-1 | hashicorp | 154 M |
| Transaction Summary |           |           |           |       |
| Install 1 Package |           |           |           |       |
| Total download size: 154 M |           |           |           |       |
| Installed size: 436 M |           |           |           |       |
| Downloading Packages: |           |           |           |       |
| vault-1.18.1-1.x86_64.rpm |           |           |           |       |
| Total |           |           |           |       |
| Hashicorp Stable - x86_64 |           |           |           |       |
| Importing GPG key 0x62E1E01: |           |           |           |       |
| Userid : HashiCorp Stable (HashiCorp Package Signing) <security+packaging@hashicorp.com> |           |           |       |
| Fingerprint: 798A E0C1 4E5C 1542 8CBE 42EE AA16 FCBC A621 E701 |           |           |       |
| From: https://releases.hashicorp.com/gpg |           |           |           |       |
| Key imported successfully |           |           |           |       |
| Running transaction check |           |           |           |       |
| Transaction check succeeded. |           |           |           |       |
| Running transaction test |           |           |           |       |
| Transaction test succeeded. |           |           |           |       |
| Running transaction |           |           |           |       |
| Preparing: |           |           |           |       |
| Running scriptlet: vault-1.18.1-1.x86_64 |           |           |       |
| Installed: |           |           |           |       |
| vault-1.18.1-1.x86_64 |           |           |       |
| Complete! |           |           |           |       |
[root@ip-172-31-42-27 ~]#
```

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
  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
  secrets  Interact with secrets engines
  ssh      Initiate an SSH session
  token
  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/ytj5ESqD4wBSuJ7EIIKvgI6TpV31U8KqewN0EwnI=
Root Token: hvs.P106ZCFFnhNW453CxuGcbhbl

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
   _#
  /_###_
 /_\#\#\#\_
 \#\#\#
  \#/ ___ 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 ~]# export VAULT_ADDR='http://127.0.0.1:8200'  
[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**

The screenshot shows the IAM Dashboard with the following details:

- Identity and Access Management (IAM)** sidebar with options like User groups, Users, Roles, Policies, Identity providers, Account settings, and Root access management.
- IAM Dashboard Info** section with **Security recommendations**:
 - Root user has MFA (Green checkmark)
 - Root user has no active access keys (Green checkmark)
- IAM resources** section showing counts:

User groups	Users	Roles	Policies	Identity providers
1	2	12	1	0

Click on **Create policy**

The screenshot shows the Policies page with the following details:

- Identity and Access Management (IAM)** sidebar with options like User groups, Users, Roles, Policies, Identity providers, Account settings, and Root access management.
- Policies (1284) Info** section with a search bar and filter dropdown set to "All types".
- A list of policies:

Policy name	Type	Used as	Description
AccessAnalyzerServiceRolePolicy	AWS managed	None	Allow Access Analyzer to analyze resou...
AdministratorAccess	AWS managed - job function	Permissions policy (1)	Provides full access to AWS services an...
AdministratorAccess-Amplify	AWS managed	None	Grants account administrative permis...
AdministratorAccess-AWSElasticBeanstalk	AWS managed	None	Grants account administrative permis...
AlexaForBusinessDeviceSetup	AWS managed	None	Provide device setup access to AlexaFo...
AlexaForBusinessFullAccess	AWS managed	None	Grants full access to AlexaForBusiness ...

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

```
1  {
2    "Version": "2012-10-17",
3    "Statement": [
4      {
5        "Effect": "Allow",
6        "Action": "sts:AssumeRole",
7        "Resource": "arn:aws:iam::116981809238:role/VaultCrossAccountRole"
8      }
9    ]
10 }
```

- 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**

The screenshot shows the 'Specify permissions' step of the 'Create policy' wizard. The 'Policy editor' section contains the following JSON code:

```
1  {
2    "Version": "2012-10-17",
3    "Statement": [
4      {
5        "Effect": "Allow",
6        "Action": "sts:AssumeRole",
7        "Resource": "arn:aws:iam::116981809238:role/VaultCrossAccountRole"
8      }
9    ]
10 }
11 }
```

Below the code, there are buttons for '+ Add new statement' and 'Edit statement'. To the right, there's a sidebar titled 'Select a statement' with a button '+ Add new statement'. At the bottom, it shows '5995 of 6144 characters remaining' and status indicators: Security: 0, Errors: 0, Warnings: 0, Suggestions: 0. The 'Actions' tab is selected at the top right.

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

The screenshot shows the 'Review and create' step. It includes sections for 'Policy details' (with fields for Policy name and Description), 'Permissions defined in this policy' (listing 'STS Limited: Write' with 'RoleName| string like |VaultCrossAccountRole'), and 'Add tags - optional' (with a note about key-value pairs). The 'Create policy' button is visible at the bottom right.

Now Policy has been created

The screenshot shows the 'Policies' list. A green banner at the top says 'Policy VAULT_CROSS_ACCOUNT_POLICY created.' The list shows two policies: 'AccessAnalyzerServiceRolePolicy' (AWS managed, None used as) and 'AdministratorAccess' (AWS managed - job function, Permissions policy (1)). The 'Create policy' button is visible at the top right.

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



IAM > Roles

Identity and Access Management (IAM)

Dashboard

Access management

User groups

Users

Roles

Policies

Identity providers

Account settings

Root access management New

Access reports

Roles (12) Info

An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.

Role name	Trusted entities	Last activity
AWSDefaultRecycleManagerDefaultRole	AWS Service: dlm	25 days ago
AWSServiceRoleForAmazonEKS	AWS Service: eks (Service-Linked Role)	3 days ago
AWSServiceRoleForAmazonEKSForFargate	AWS Service: eks-fargate (Service-Link)	3 days ago
AWSServiceRoleForAmazonEKSNodegroup	AWS Service: eks-nodegroup (Service-Link)	3 days ago
AWSServiceRoleForAutoScaling	AWS Service: autoscaling (Service-Link)	3 days ago
AWSServiceRoleForEC2Spot	AWS Service: spot (Service-Linked Role)	-
AWSServiceRoleForElasticLoadBalancing	AWS Service: elasticloadbalancing (Service-Linked Role)	4 days ago
AWSServiceRoleForRDS	AWS Service: rds (Service-Linked Role)	1 hour ago

Search

Delete Create role

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

Select trusted entity Info

Trusted entity type

AWS service

Allow AWS services like EC2, Lambda, or others to perform actions in this account.

AWS account

Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.

Web Identity

Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.

SAML 2.0 federation

Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.

Custom trust policy

Create a custom trust policy to enable others to perform actions in this account.

Use case

Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Service or use case

EC2

Choose a use case for the specified service.

Use case

EC2

Allows EC2 instances to call AWS services on your behalf.

EC2 Role for AWS Systems Manager

Allows EC2 instances to call AWS services like CloudWatch and Systems Manager on your behalf.

EC2 Spot Fleet Role

Allows EC2 Spot Fleet to request and terminate Spot Instances on your behalf.

EC2 - Spot Fleet Auto Scaling

Allows Auto Scaling to access and update EC2 spot fleets on your behalf.

EC2 - Spot Fleet Tagging

Allows EC2 to launch spot instances and attach tags to the launched instances on your behalf.

EC2 - Spot Instances

Allows EC2 Spot Instances to launch and manage spot instances on your behalf.

EC2 - Spot Fleet

Allows EC2 Spot Fleet to launch and manage spot fleet instances on your behalf.

EC2 - Scheduled Instances

Allows EC2 Scheduled Instances to manage instances on your behalf.

Cancel Next

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

Add permissions Info

Permissions policies (1/997) Info

Choose one or more policies to attach to your new role.

Filter by Type

Q: Vault

All types

2 matches

Policy name	Type	Description
Vault_Cross_Account_Policy	Customer managed	-
vault-dynamic-creds	Customer managed	-

Set permissions boundary - optional

Cancel Previous Next

Enter the role name and click Create role

Name, review, and create

Role details

Role name

Enter a meaningful name to identify this role.

Vault_server_role

Maximum 64 characters. Use alphanumeric and '+'-'_.' characters.

Description

Add a short explanation for this role.

Allows EC2 instances to call AWS services on your behalf.

Maximum 1000 characters. Use letters (A-Z and a-z), numbers (0-9), tabs, new lines, or any of the following characters: '_+=.,@/\[\{\}\]#\$%^&`~-`'

Create role



The IAM role has now been successfully created.

Roles (13) Info		
	Role name	Trusted entities
<input type="checkbox"/>	AWSDataLifecycleManagerDefaultRole	AWS Service: dlm 25 days ago
<input type="checkbox"/>	AWSServiceRoleForAmazonEKS	AWS Service: eks (Service-Linked Rol 3 days ago)
<input type="checkbox"/>	AWSServiceRoleForAmazonEKSFargate	AWS Service: eks-fargate (Service-Li 3 days ago)
<input type="checkbox"/>	AWSServiceRoleForAmazonEKSNodegroup	AWS Service: eks-nodegroup (Servic 3 days ago)
<input type="checkbox"/>	AWSServiceRoleForAutoScaling	AWS Service: autoscaling (Service-Li 3 days ago)
<input type="checkbox"/>	AWSServiceRoleForEC2Spot	AWS Service: spot (Service-Linked R 23 minutes ago)
<input type="checkbox"/>	AWSServiceRoleForElasticLoadBalancing	AWS Service: elasticloadbalancing (S 4 days ago)
<input type="checkbox"/>	AWSServiceRoleForRDS	AWS Service: rds (Service-Linked Rol 23 minutes ago)
<input type="checkbox"/>	AWSServiceRoleForSupport	AWS Service: support (Service-Linker -)

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**

The screenshot shows the AWS IAM Dashboard. On the left sidebar, under 'Access management', 'Policies' is selected. In the main area, there's a 'Security recommendations' box with two items: 'Add MFA for root user' and 'Root user has no active access keys'. Below it is an 'IAM resources' box showing 0 User groups, 0 Users, 2 Roles, 0 Policies, and 0 Identity providers. To the right, there's an 'AWS Account' box with account details (Account ID: 116981809238, Account Alias: Create, Sign-in URL: https://116981809238.sigin.aws.amazon.com/console) and a 'Quick Links' box for managing security credentials.

Click on **Create policy**

The screenshot shows the 'Policies (1283)' list page. At the top, there's a search bar, a 'Filter by Type' dropdown set to 'All types', and a 'Create policy' button. The main table lists 1283 policies, each with a checkbox, a preview icon, the policy name, its type (e.g., 'AWS managed'), and the number of 'Used as'. To the right of the table, there's a column for 'Description' with links to each policy's details.

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



```
1  {
2      "Version": "2012-10-17",
3      "Statement": [
4          {
5              "Effect": "Allow",
6              "Action": [
7                  "s3>ListBucket",
8                  "s3GetObject",
9                  "ec2DescribeInstances"
10             ],
11             "Resource": "*"
12         }
13     ]
14 }
```

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

Specify permissions Info
Add permissions by selecting services, actions, resources, and conditions. Build permission statements using the JSON editor.

Policy editor

```
1 * {
2     "Version": "2012-10-17",
3     "Statement": [
4         {
5             "Effect": "Allow",
6             "Action": [
7                 "s3:listBucket",
8                 "s3:GetObject",
9                 "ec2:DescribeInstances"
10            ],
11            "Resource": "*"
12        }
13    ]
14 }
```

Visual **JSON** Actions ▾

Edit statement

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

+ Add new statement

+ Add new statement

JSON | Ln 15, Col 0 6006 of 6144 characters remaining

Cancel Next

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



Review and create Info
Review the permissions, specify details, and tags.

Policy details

Policy name
Enter a meaningful name to identify this policy.

vault_s3_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.

Permissions defined in this policy Info

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

Search

[Edit](#)

[Show remaining 426 services](#)

Allow (2 of 428 services)

Service	Access level	Resource	Request condition
EC2	Limited: List	All resources	None
S3	Limited: List, Read	All resources	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.

[Add new tag](#)

You can add up to 50 more tags.

[Cancel](#)

[Previous](#)

[Create policy](#)

The policy has been successfully created with S3 and EC2 access

Policy vault_s3_policy created.			
View policy X			
Actions Delete Create policy			
Policies (1284) <small>Info</small> A policy is an object in AWS that defines permissions.			
Policy name	Type	Used as	Description
<input checked="" type="checkbox"/> AccessAnalyzerServiceRolePolicy	AWS managed	None	Allow Access Analyzer to analyze resou...
<input checked="" type="checkbox"/> AdministratorAccess	AWS managed - job function	None	Provides full access to AWS services an...
<input checked="" type="checkbox"/> AdministratorAccess-Amplify	AWS managed	None	Grants account administrative permis...
<input checked="" type="checkbox"/> AdministratorAccess-AWSElasticBeanstalk	AWS managed	None	Grants account administrative permis...
<input checked="" type="checkbox"/> AlexaForBusinessDeviceSetup	AWS managed	None	Provide device setup access to AlexaFo...

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

Roles (2) <small>Info</small>			
Actions Delete Create role			
Role name	Trusted entities	Last activity	Description
<input type="checkbox"/> AWSServiceRoleForSupport	AWS Service: support (Service-Linked)	-	Allow Access Analyzer to analyze resou...
<input type="checkbox"/> AWSServiceRoleForTrustedAdvisor	AWS Service: trustedadvisor (Service)	-	Provides full access to AWS services an...

Roles Anywhere Info

Authenticate your non AWS workloads and securely provide access to AWS services.

[Access AWS from your non AWS workloads](#)

Operate your non AWS workloads using the same authentication and authorization strategy that you use within AWS.

X.509 Standard

Use your own existing PKI infrastructure or use [AWS Certificate Manager Private Certificate Authority](#) to authenticate identities.

Temporary credentials

Use temporary credentials with ease and benefit from the enhanced security they provide.

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



Select trusted entity [Info](#)



Trusted entity type

- AWS service Allow AWS services like EC2, Lambda, or others to perform actions in this account.
- AWS account Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.
- Web identity Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.
- SAML 2.0 federation Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.
- Custom trust policy Create a custom trust policy to enable others to perform actions in this account.

An AWS account

Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.

This account (116981809238)

Another AWS account

Account ID

Identifier of the account that can use this role

038462791702

Account ID is a 12-digit number.

Options

Require external ID (Best practice when a third party will assume this role)

Require MFA Requires that the assuming entity use multi-factor authentication.

[Cancel](#)

[Next](#)

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

Add permissions [Info](#)

Permissions policies (1/996) [Info](#)

Choose one or more policies to attach to your new role.

Policy name	Type	Description
vault	All types	Customer managed
<input checked="" type="checkbox"/> vault_s3_policy		

► Set permissions boundary - optional

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

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

Name, review, and create

Role details

Role name Enter a meaningful name to identify this role.

Description Add a short description for this role.

Step 1: Select trusted entities

Trust policy

```
1: {
2:     "Version": "2012-10-17",
3:     "Statement": [
4:         {
5:             "Effect": "Allow",
6:             "Action": "sts:AssumeRole",
7:             "Principal": "arn:aws:iam::038462791702",
8:             "Condition": {}
9:         }
10:    ]
11: }
```

Step 2: Add permissions

Permissions policy summary

Policy name	Type	Attached as
vault_s3_policy	Customer managed	Permissions policy

Step 3: Add tags

Add tags - optional [Info](#)

No tags are currently attached to this role.

No tags associated with the resource.

[Add new tag](#)

You can add up to 50 more tags.

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

The role has been successfully created



The screenshot shows the AWS IAM Roles page. A green banner at the top indicates that a role has been created. Below this, a table lists roles with columns for Role name, Trusted entities, and Last activity. The 'VaultCrossAccountRole' is listed under 'Role name'. Under 'Trusted entities', it shows 'AWS Service support [Service-Links]' and 'AWS Service trustedadvisor [Service]'. The 'Last activity' column shows 'Account: 038462791702'. At the bottom, there are sections for 'Roles Anywhere' (AWS Lambda), 'X.509 Standard' (AWS Certificate Manager Private Certificate Authority), and 'Temporary credentials'.

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.P106ZCfFnhNW453CxuGcbhbl
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.P106ZCfFnhNW453CxuGcbhbl
token_accessor	fHBbhWLQ4C9WEOLW4BcaA623
token_duration	∞
token_renewable	false
token_policies	["root"]
identity_policies	[]
policies	["root"]

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



```
for verbose messaging see aws.com/vg/credentials

[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

```
Success! Enabled the AWS secrets engine at: aws/
[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/u0vX2j7TDnVjoCESd12FcZo4
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    3wNZumkSFs95hqbATrGtID5hk9e/XEpNFo1fdTm7
security_token FwoGZXIvYXdzEBMaDFVVh8NUbr9Hf/8n0lFaasIBndt0KwZQvQtIM0jXCaQxnPZmAUhRvtXMM0dXh5KFin2EFntBdisJENdLBEB01IwtGE2dQDEOB4wqXjhY1QSiPJ0z6tLjM4k7rUgb8Yhd
MzrrnGax7KGo5Xzb0H7wFNgsFWzdhqBbCsDrEjANqRYgxU0eMTH5vFjljxBDE4pCkoCeZntMQ10okV8Hk1E6pTs5618GzvvnDXhmAaaAyuuJJBqs3uGVevQEFNFYHnckJf5NzQfnrtEavGILlApf+Ork68xT3FPA6J18zxH
ShsLfzbYT2BmSn3y6wr4mMo/6b6uYyLx045D5A11U8gwGLH9k26VV9j/KAEDykLaPohjNIRvHuKChtrngxfvy6fWnEg==
session_token FwoGZXIvYXdzEBMaDFVVh8NUbr9Hf/8n0lFaasIBndt0KwZQvQtIM0jXCaQxnZnAUhRvtXMM0dXh5KFin2EFntBdisJENdLBEB01IwtGE2dQDEOB4wqXjhY1QSiPJ0z6tLjM4k7rUgb8Yhd
MzrrnGax7KGo5Xzb0H7wFNgsFWzdhqBbCsDrEjANqRYgxU0eMTH5vFjljxBDE4pCkoCeZntMQ10akV8Hk1E6pTs5618GzvvnDXhmAaaAyuuJJBqs3uGVevQEFNFYHnckJf5NzQfnrtEavGILlApf+Ork68xT3FPA6J18zxH
ShsLfzbYT2BmSn3y6wr4mMo/6b6uYyLx045D5A11U8gwGLH9k26VV9j/KAEDykLaPohjNIRvHuKChtrngxfvy6fWnEg==
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
```



wZQvQt1MOjXCaQxnPZmAUhRvtXMMOdXh5KFin2EFntBdisJENdLBEB0ilWtGE2dQ
DEOBQ4wqXjhY1QSiPJDz6tkLjM4k7rUgb8YhdMzrnGAx7KGo5XZb0H7wFNGsFWzdh
qBbCsDfeJANqRYGxU0eMTH5vfJLjxBDE4pCkoCeZntMQ1OakV8Hk1E6pTs5618Gzvn
vDXhmAaaAyuuJJBqs3uGVevQEFNfYHNckJf5NzQFntEAvGLltApf+Ork68xT3FPA6Jl8
zxHShslfbYTZBmSn3y6wr4mMo/6b6uQYyLXo45DSA1IU8gwGLH9kz6VVG9u/KAED
ykLAPOhjNIRvhWuKChngxfvy6fWnEg==

```
mohendranelvakumar@Mahendrans-MBP ~ % export AWS_ACCESS_KEY_ID=ASIAWRPF1YBLETFUY7JD  
mohendranelvakumar@Mahendrans-MBP ~ % export AWS_SECRET_ACCESS_KEY=3oWzUmSFs95qoATnGtIDSHk9e/XEpN0f1fdTm7  
mohendranelvakumar@Mahendrans-MBP ~ % export AWS_SESSION_TOKEN=TwOGZ1vYXzeB6M0dfVVh8Nbr9HF8m0L1Aa5IBnD0KwzQvQt1M0jXcQxnPZmAUhRvtXMMOdXh5KFin2EFntBdisJENdLBEB0l1WGE2dQ0EOBQ4wqXjhY1Q51PJ1Dz6tkLjM4k7rUgb8YhdMzrhGx7KGo5XZb0H7wFNGsFWzdhqBbCsDfeJANqRYGxU0eMTH5vfJLjxBDE4pCkoCeZntMQ1OakV8Hk1E6pTs5618Gzvnv0XhmAaaAyuuJ1Bqs3uGVevQEFNfYHNckJf5NzQFntEAvGLltApf+Ork68xT3FPA6Jl8zxHShslfbYTZBmSn3y6wr4mMo/6b6uQYyLXo45DSA1IU8gwGLH9kz6VVG9u/KAED  
ykLAPOhjNIRvhWuKChngxfvy6fWnEg==  
mohendranelvakumar@Mahendrans-MBP ~ % aws sts get-caller-identity  
{  
    "UserId": "AROAMNPFF1BLIMWXF4CN:vault-root-s3_bucket_access-1732154239-7j3DLjVfMHmbM9aGUu4u",  
    "Account": "116981809238",  
    "Arn": "arn:aws:sts::116981809238:assumed-role/VaultCrossAccountRole/vault-root-s3_bucket_access-1732154239-7j3DLjVfMHmbM9aGUu4u"  
}  
mohendranelvakumar@Mahendrans-MBP ~ %
```

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

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

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/>