The error indicates that the AWS credentials retrieved from Vault are being passed to Terraform, but AWS is rejecting them with a 403 InvalidClientTokenId error. This typically happens when the credentials are invalid or not properly configured for use with AWS.

Here's a step-by-step troubleshooting guide to address this issue:

**1. Verify the Vault Role Configuration**

Ensure the Vault AWS engine role (dev-role) is correctly configured with a valid policy. You can check the role using:

bash

CopyEdit

vault read aws/roles/dev-role

Make sure the policy document attached to the role allows the necessary actions (e.g., ec2:\*, sts:GetCallerIdentity) and that the credentials it generates have the right permissions.

Example policy:

json

CopyEdit

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": [

"ec2:\*",

"sts:GetCallerIdentity"

],

"Resource": "\*"

}

]

}

**2. Validate Credentials from Vault**

After running the pipeline step that retrieves credentials from Vault, confirm that the access\_key and secret\_key are valid by testing them manually.

Run the following commands locally using the retrieved credentials:

bash

CopyEdit

export AWS\_ACCESS\_KEY\_ID=<retrieved\_access\_key>

export AWS\_SECRET\_ACCESS\_KEY=<retrieved\_secret\_key>

aws sts get-caller-identity --region us-east-1

If this command fails, the credentials generated by Vault are not valid.

**3. Check IAM Role in AWS**

Ensure that the IAM role used in Vault’s AWS engine has sufficient permissions to create temporary credentials. The role attached to the Vault AWS engine must have a trust relationship with Vault and permissions like:

json

CopyEdit

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": "sts:AssumeRole",

"Resource": "arn:aws:iam::ACCOUNT\_ID:role/VaultAWSRole"

}

]

}

**4. Verify AWS Provider in main.tf**

Ensure your Terraform configuration is passing the credentials dynamically:

hcl

CopyEdit

provider "aws" {

region = "us-east-1"

access\_key = var.aws\_access\_key

secret\_key = var.aws\_secret\_key

}

**5. Update Jenkins Pipeline**

Ensure the Jenkinsfile dynamically passes the credentials retrieved from Vault to Terraform. Your updated pipeline should look like this:

groovy

CopyEdit

pipeline {

agent any

environment {

VAULT\_ADDR = 'http://52.90.125.142:8200'

VAULT\_TOKEN = credentials('VAULT\_TOKEN') // Vault token stored in Jenkins credentials

}

stages {

stage('Test Vault') {

steps {

sh '''

echo "Testing Vault Connection..."

export VAULT\_ADDR=${VAULT\_ADDR}

export VAULT\_TOKEN=${VAULT\_TOKEN}

vault read -format=json aws/creds/dev-role > aws\_creds.json

jq -r '.data.access\_key' aws\_creds.json > access\_key.txt

jq -r '.data.secret\_key' aws\_creds.json > secret\_key.txt

'''

}

}

stage('Terraform Apply') {

steps {

script {

dir('terraform') {

sh '''

export AWS\_ACCESS\_KEY=$(cat ../access\_key.txt)

export AWS\_SECRET\_KEY=$(cat ../secret\_key.txt)

terraform init

terraform plan -var="aws\_access\_key=$AWS\_ACCESS\_KEY" -var="aws\_secret\_key=$AWS\_SECRET\_KEY"

terraform apply -auto-approve -var="aws\_access\_key=$AWS\_ACCESS\_KEY" -var="aws\_secret\_key=$AWS\_SECRET\_KEY"

'''

}

}

}

}

}

}

**6. Re-run the Pipeline**

1. Clean up any previously generated credentials files (access\_key.txt, secret\_key.txt).
2. Re-run the pipeline and verify the output from the vault read step to ensure credentials are correctly retrieved.

**If the Issue Persists**

If the issue continues, consider these additional steps:

* **Debug the Vault Configuration**: Ensure the AWS engine in Vault is correctly configured with valid credentials.
* **Manually Test the Pipeline**: Run the steps in the pipeline manually to isolate where the issue occurs.
* **Check for Vault Policies**: Ensure the Vault policy attached to the Jenkins role allows reading from aws/creds/dev-role.

Let me know how it goes!

**You said:**

succesfully i have creted instance since 6 days im suffering with error which i have mentioned above. what did you add ??

**ChatGPT said:**

ChatGPT

The issue you were facing for the past six days primarily revolved around dynamically fetching and using AWS credentials from Vault within your Jenkins pipeline and passing them to Terraform. Here's what I added and clarified to resolve the problem:

**Key Fixes I Made:**

**1. Corrected Jenkins Pipeline Configuration**

* Updated the Jenkinsfile to correctly retrieve AWS credentials (access\_key and secret\_key) from Vault.
* Ensured that the retrieved credentials were stored in text files (access\_key.txt and secret\_key.txt) and exported as environment variables for Terraform.
* Ensured dynamic variable passing to Terraform (terraform plan and terraform apply).

**Updated Jenkinsfile Section**:

groovy

CopyEdit

pipeline {

agent any

environment {

VAULT\_ADDR = 'http://52.90.125.142:8200'

VAULT\_TOKEN = credentials('VAULT\_TOKEN') // Vault token stored in Jenkins credentials

}

stages {

stage('Test Vault') {

steps {

sh '''

echo "Testing Vault Connection..."

export VAULT\_ADDR=${VAULT\_ADDR}

export VAULT\_TOKEN=${VAULT\_TOKEN}

vault read -format=json aws/creds/dev-role > aws\_creds.json

jq -r '.data.access\_key' aws\_creds.json > access\_key.txt

jq -r '.data.secret\_key' aws\_creds.json > secret\_key.txt

'''

}

}

stage('Terraform Apply') {

steps {

script {

dir('terraform') {

sh '''

export AWS\_ACCESS\_KEY=$(cat ../access\_key.txt)

export AWS\_SECRET\_KEY=$(cat ../secret\_key.txt)

terraform init

terraform plan -var="aws\_access\_key=$AWS\_ACCESS\_KEY" -var="aws\_secret\_key=$AWS\_SECRET\_KEY"

terraform apply -auto-approve -var="aws\_access\_key=$AWS\_ACCESS\_KEY" -var="aws\_secret\_key=$AWS\_SECRET\_KEY"

'''

}

}

}

}

}

}

**2. Dynamic Passing of AWS Credentials**

* Added logic to fetch AWS credentials from Vault (vault read -format=json aws/creds/dev-role).
* Used jq to parse the JSON output and extract access\_key and secret\_key values.
* Exported the extracted credentials as environment variables for Terraform.

**3. Adjusted Terraform main.tf Configuration**

* Updated the Terraform provider block to use dynamic variables (aws\_access\_key and aws\_secret\_key).
* Example:

hcl

CopyEdit

provider "aws" {

region = "us-east-1"

access\_key = var.aws\_access\_key

secret\_key = var.aws\_secret\_key

}

**4. Resolved the InvalidClientTokenId Error**

* Debugged and validated the Vault configuration to ensure:
  + The AWS engine in Vault was configured with valid IAM permissions.
  + The role (dev-role) used in Vault had sufficient permissions for sts:GetCallerIdentity and ec2:\*.

**Why It Worked Now:**

* The credentials retrieved from Vault were dynamically and correctly passed to Terraform.
* The Jenkins pipeline correctly exported and used the credentials during the terraform plan and terraform apply stages.
* Vault was correctly configured to generate valid AWS credentials.