Deploy Private EKS Terraform Code using GitHub Actions with OpenID Connect (OIDC) identity providers (AWS Cloud)

This blog guides integrating OIDC with GitHub and AWS. This integration enhances security by removing the necessity for persistent IAM access keys, offering a safer approach for accessing AWS resources directly from your GitHub repository.

Securing EKS Resource Deployment: Integrating OIDC IAM Roles with GitHub Actions:

OpenID Connect (OIDC) allows your GitHub Actions workflows to access resources in Amazon Web Services (AWS), without needing to store the AWS credentials as long-lived GitHub secrets.

Benefits of using OIDC

- No cloud secrets: You won't need to duplicate your cloud credentials as long-lived GitHub secrets. Instead, you can configure the OIDC trust on your cloud provider, and then update your workflows to request a short-lived access token from the cloud provider through OIDC.
- Authentication and authorization management: You have more granular control over how workflows can use credentials, using your cloud provider's authentication (authN) and authorization (authZ) tools to control access to cloud resources.
- Rotating credentials: With OIDC, your cloud provider issues a short-lived access token that is only valid for a single job, and then automatically expires.

This guide explains how to configure AWS to trust GitHub's OIDC as a federated identity, and includes a workflow example for the aws-actions/configure-aws-credentials that uses tokens to authenticate to AWS and access resources.

Prerequisites:

The first step in this process is to create an OIDC provider which you will use in the trust policy for the IAM role used in the Github actions.

- This role is used by Terraform to provision and manage your EKS cluster.
- It should have sufficient permissions to create and manage EKS resources.

To Initiate the Deployment Process, pre-requisites for EKS Cluster Terraform Provisioning:

- Use of Cloud formation stack present at location**: /devops-assessment/cloudformation/oidc-infra-template.yml will configure following**
  - **Name of Github user** : "muneebbukhari555"
  - **GitHub Repo name**: " * " (Using Wildcard for other Repo)
  - **OIDC Provider**: **token.actions.githubusercontent.com**
  - **RoleName:** GitHub_Actions_Infra_Role
  - **S3 Bucket**: rak-terraform-iaac-eks (Versioning enable)
  - **DynamoDB Table:** prod-rak-eksdemo

Update the below command as per above inputs:

```
aws cloudformation deploy \
    --stack-name OidcInfraTemplateStack \
    --template-file oidc-infra-template.yml \
    --capabilities "CAPABILITY_IAM" "CAPABILITY_NAMED_IAM" \
    --parameter-overrides \
    RepositoryName='muneebbukhari555/*' \
```

```
    BucketName='rak-terraform-iaac-eks'\
    DynamoTableName='prod-rak-eksdemo'
```

Once Stack is deployed. You will see four different things that have been configured:

- s3 Bucket
- DynamoDB Table
- OIDC Github Provider on AWS
- Infra Execution Role with Admin policy attached to provision EKS cluster.

> S3 Bucket & DynamoDB Table for terraform remote state & locking purpose.
> Later we will use these details in our terraform EKS Deployment.

Clone repo from Github on your local machine:

Repo Link: https://github.com/muneebbukhari555/devops-assessment

We can use above role created earlier for AWS authentication and can use workflow present at: **/devops-assessment/.github/workflows/infra-provision.yml**

Deploy Infra Workflow Pipeline:

Github Workflow that will create a EKS Cluster Deployment, After adding the environment, and secret variables in the repository. We can deploy the resource on the cluster using the IAM role created earlier.

> **Switched from main to feature branch**
>
> We don't want push our terraform code into main branch Directly that's why first push code into feature branch then Merage into main branch

By taking reference from Cloudformation Stack and use Repository Secrets and Environment Variable required in the Worflow:

- AWS_REGION
- AWS_ACCOUNT_ID
- TF_STATE_S3_BUCKET
- TF_State_File_Key
- TF_STATE_DYNAMODB_TABLE

Trigger Condition:

Branch: GitHub Action Trigger always run from “main” branch.

Paths: only changes done from "terraform-eks/rak-eks-prod-cluster/*”

Destroy Infra Workflow:

For Destroy EKS Cluster we can use same stretegy here. We can use above role created earlier for AWS authentication and can use workflow present at: **/devops-assessment/.github/workflows/infra-destroy.yml**

Trigger Condition:

No Branch: GitHub Action Trigger always run Manually.