Practical-11

Option A: OIDC

OIDC lets GitHub connect to AWS without storing access keys.

It gives temporary, secure credentials during workflow runs.

Steps

1. In AWS IAM → add GitHub OIDC provider:
   https://token.actions.githubusercontent.com
2. Create an IAM Role with trust policy for GitHub.
3. Attach required permissions (example: S3, DynamoDB, Terraform).
4. Use the role in GitHub workflow.

Example GitHub Workflow Using OIDC

```yaml
name: Deploy

on:
  push:
    branches: [ "main" ]

jobs:
  deploy:
    runs-on: ubuntu-latest
    permissions:
      id-token: write
      contents: read

    steps:
      - uses: actions/checkout@v4

      - name: Configure AWS credentials (OIDC)
        uses: aws-actions/configure-aws-credentials@v4
        with:
          role-to-assume: arn:aws:iam::123456789012:role/GitHubOIDCRole
          aws-region: ap-south-1

      - name: Terraform Apply
        run: terraform apply -auto-approve
```

No access keys used → safer.

Option B: Access Keys Method

In this method, GitHub uses IAM user access keys stored in GitHub Secrets.

Steps

Create IAM user with “Programmatic Access”.

Copy:

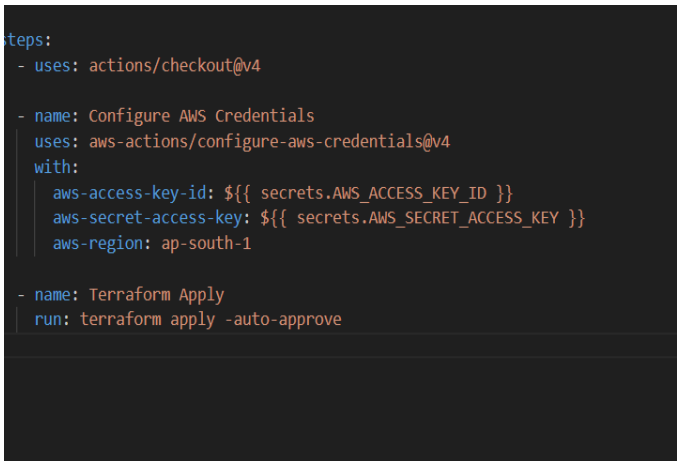
AWS_ACCESS_KEY_ID

AWS_SECRET_ACCESS_KEY

Add them in GitHub → Settings → Secrets → Actions.

Use them in workflow.

Example Workflow (Access Keys)



```yaml
steps:
 - uses: actions/checkout@v4

 - name: Configure AWS Credentials
   uses: aws-actions/configure-aws-credentials@v4
   with:
     aws-access-key-id: ${{ secrets.AWS_ACCESS_KEY_ID }}
     aws-secret-access-key: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
     aws-region: ap-south-1

 - name: Terraform Apply
   run: terraform apply -auto-approve
```

Simple but less secure because keys are long-term.

Environment (prod/dev) Setup

Used to require approval before deployment.

Steps:

GitHub → Settings → Environments

Create environment name: prod

Add Required Reviewers

Example in Workflow

environment: prod

Now terraform apply will wait for review approval.