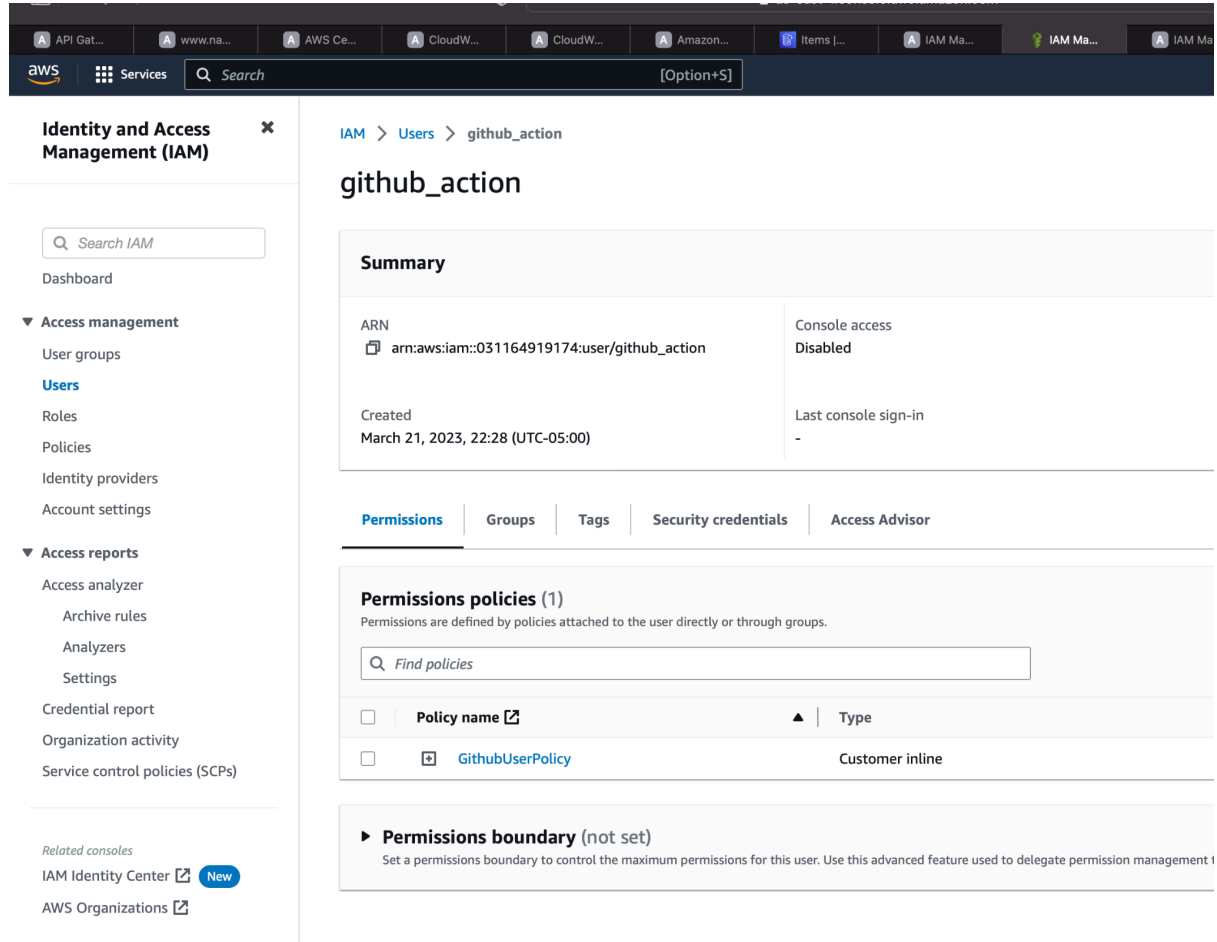


CI/CD S3 deployment with Github Actions

Step 1: Create an IAM user to be used by GitHub actions.



The screenshot shows the AWS IAM console interface. The left sidebar contains the 'Identity and Access Management (IAM)' menu with options like Dashboard, Access management, Users, Roles, Policies, Identity providers, Account settings, Access reports, and Related consoles. The main content area displays the details for the 'github_action' user. The 'Summary' section shows the ARN as 'arn:aws:iam::031164919174:user/github_action', Console access as 'Disabled', and the creation date as 'March 21, 2023, 22:28 (UTC-05:00)'. The 'Permissions' tab is selected, showing 'Permissions policies (1)' with a table listing 'GithubUserPolicy' as a 'Customer inline' policy. A 'Permissions boundary' section indicates it is 'not set'.

Identity and Access Management (IAM)

Search IAM

Dashboard

▼ Access management

- User groups
- Users**
- Roles
- Policies
- Identity providers
- Account settings

▼ Access reports

- Access analyzer
 - Archive rules
 - Analyzers
 - Settings
- Credential report
- Organization activity
- Service control policies (SCPs)

Related consoles

- [IAM Identity Center](#) **New**
- [AWS Organizations](#)

github_action

Summary

ARN arn:aws:iam::031164919174:user/github_action	Console access Disabled
Created March 21, 2023, 22:28 (UTC-05:00)	Last console sign-in -

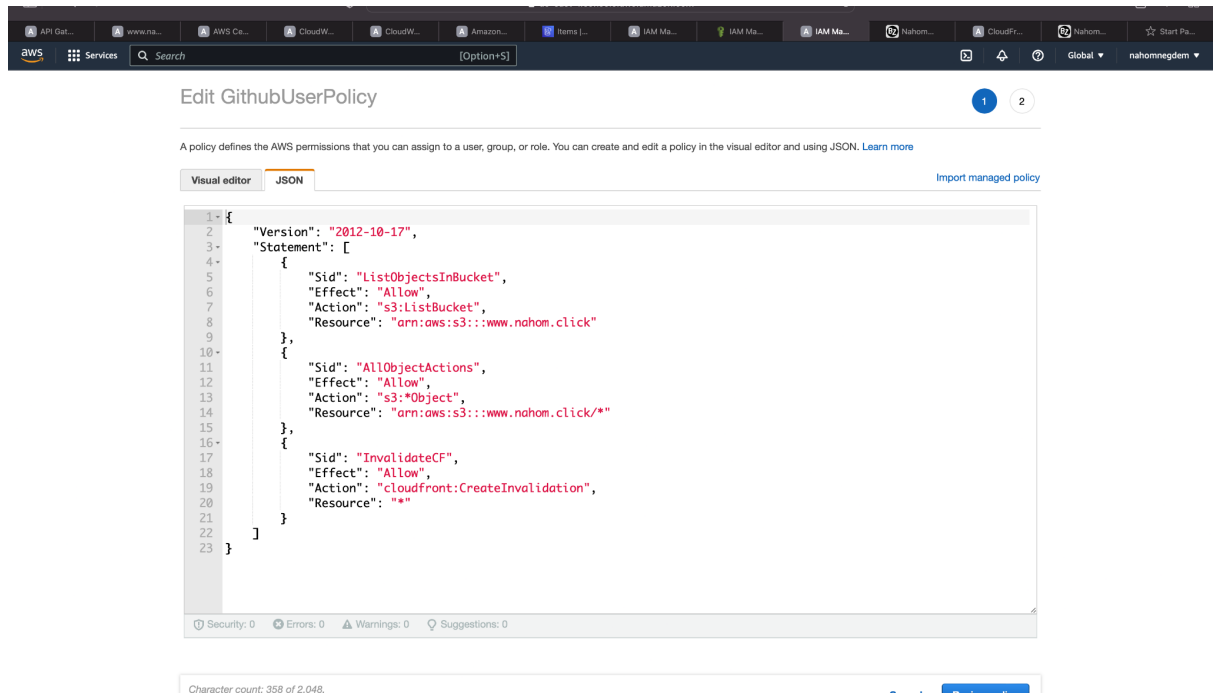
Permissions | Groups | Tags | Security credentials | Access Advisor

Permissions policies (1)
Permissions are defined by policies attached to the user directly or through groups.

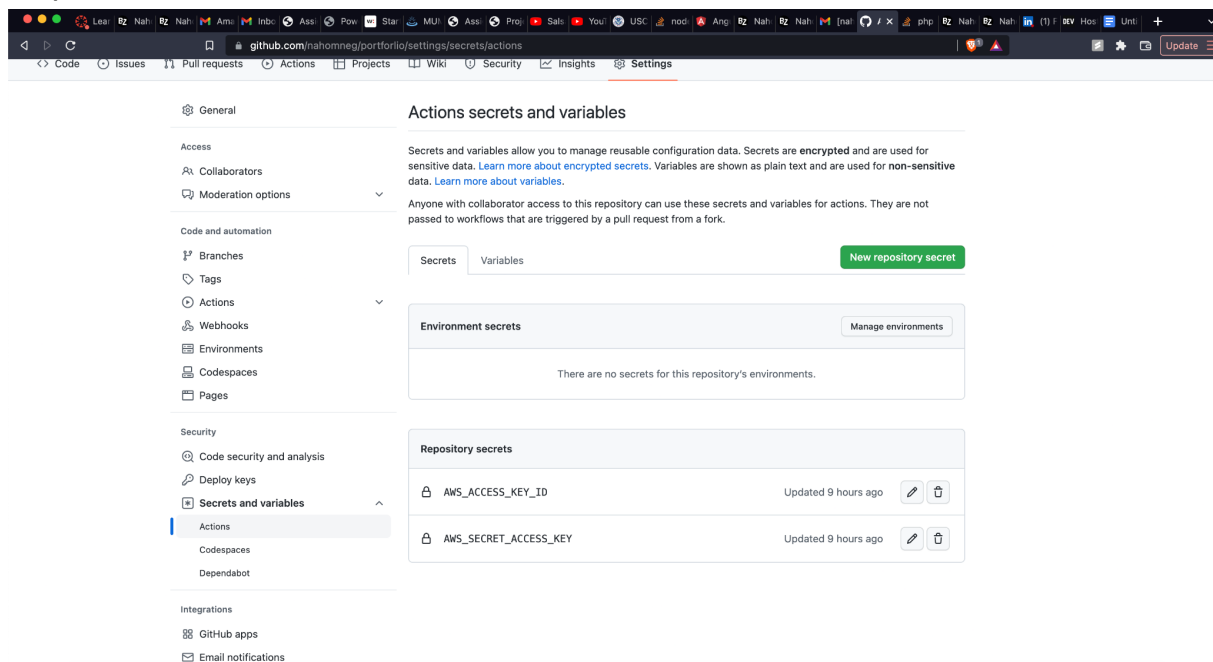
<input type="checkbox"/>	Policy name ↗	Type
<input type="checkbox"/>	GithubUserPolicy	Customer inline

► **Permissions boundary (not set)**
Set a permissions boundary to control the maximum permissions for this user. Use this advanced feature used to delegate permission management

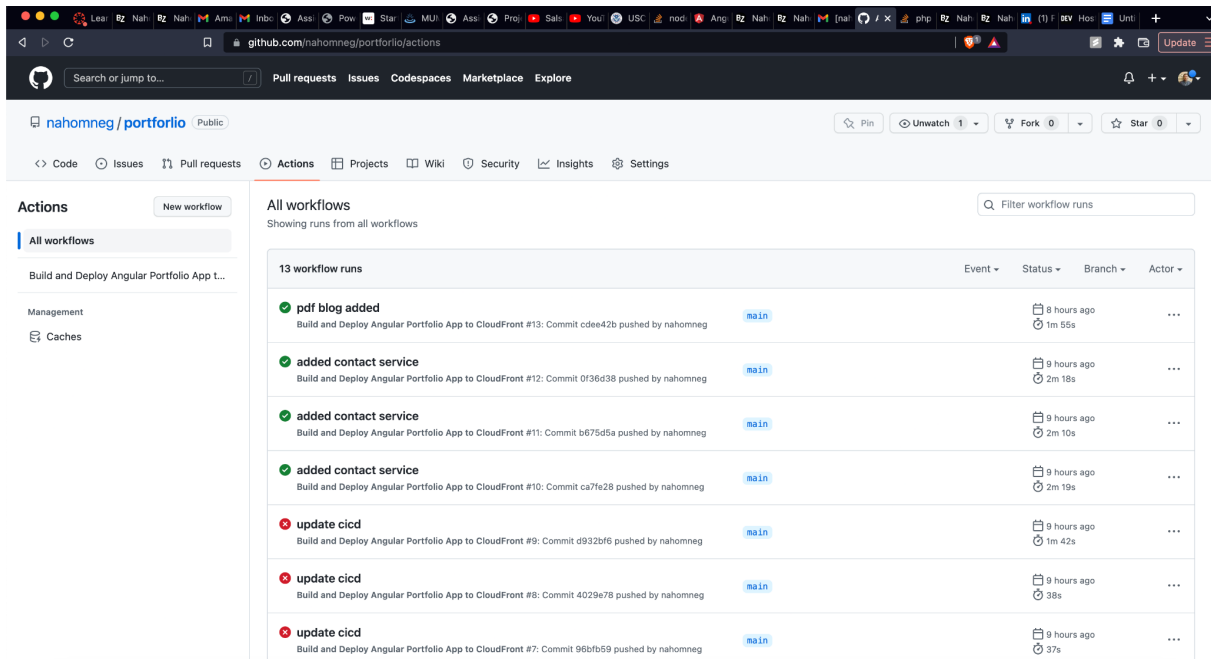
Step 2: Add the following permission by specifying the S3 bucket



Step 3: Set secret variables in Github.



Step 4: Create a workflow in github actions by clicking on new workflow button



Create a new workflow and type the following in the yaml file

```
---
name: Build and Deploy Angular Portfolio App to CloudFront
on:
  push:
    branches: [ main ]
jobs:
  build-and-deploy:
    name: Build and Deploy
    runs-on: ubuntu-latest
    env:
      BUCKET: www.nahom.click
      DIST: dist/my-portfolio
      REGION: us-east-1
      DIST_ID: E3OQ3A6YJIAJAF

    steps:
      - name: Checkout code
        uses: actions/checkout@v2

      - name: Configure AWS Credentials
        uses: aws-actions/configure-aws-credentials@v1
        with:
          aws-access-key-id: ${ secrets.AWS_ACCESS_KEY_ID }
          aws-secret-access-key: ${ secrets.AWS_SECRET_ACCESS_KEY }
          aws-region: ${ env.REGION }

      - name: Set up Node.js environment
        uses: actions/setup-node@v2
        with:
          node-version: '14'

      - name: Install Dependencies
```

```

run: |
  node --version
  npm ci --production

- name: Build Static Website
  run: |
    npm link @angular/cli
    npm install @angular-devkit/build-angular --force
    npm run build

- name: Copy files to the production website with the AWS CLI
  run: |
    aws s3 sync --delete ${ env.DIST } s3://${ env.BUCKET }

- name: Copy files to the production website with the AWS CLI
  run: |
    aws cloudfront create-invalidation \
      --distribution-id ${ env.DIST_ID } \
      --paths "/*"

```

Step 5: pull the yaml file to your local file and now you can push new changes to github and a work flow to update the s3 and invalidate the previous caches in CloudFront will happen.

The screenshot shows a GitHub Actions workflow run for the repository 'nahomneg/portfolio'. The workflow is titled 'Build and Deploy Angular Portfolio App to CloudFront' and is triggered by a pull request. The job 'Build and Deploy' has successfully completed, taking 1m 44s. The job steps are listed as follows:

- Set up job (2s)
- Checkout code (1s)
- Configure AWS Credentials (8s)
- Set up Node.js environment (8s)
- Install Dependencies (5s)
- Build Static Website (1m 27s)
- Copy files to the production website with the AWS CLI (4s)
- Copy files to the production website with the AWS CLI (1s)
- Post Set up Node.js environment (8s)
- Post Configure AWS Credentials (8s)
- Post Checkout code (8s)
- Complete job (8s)

The workflow file is located at `.github/workflows/build-and-deploy.yml`.