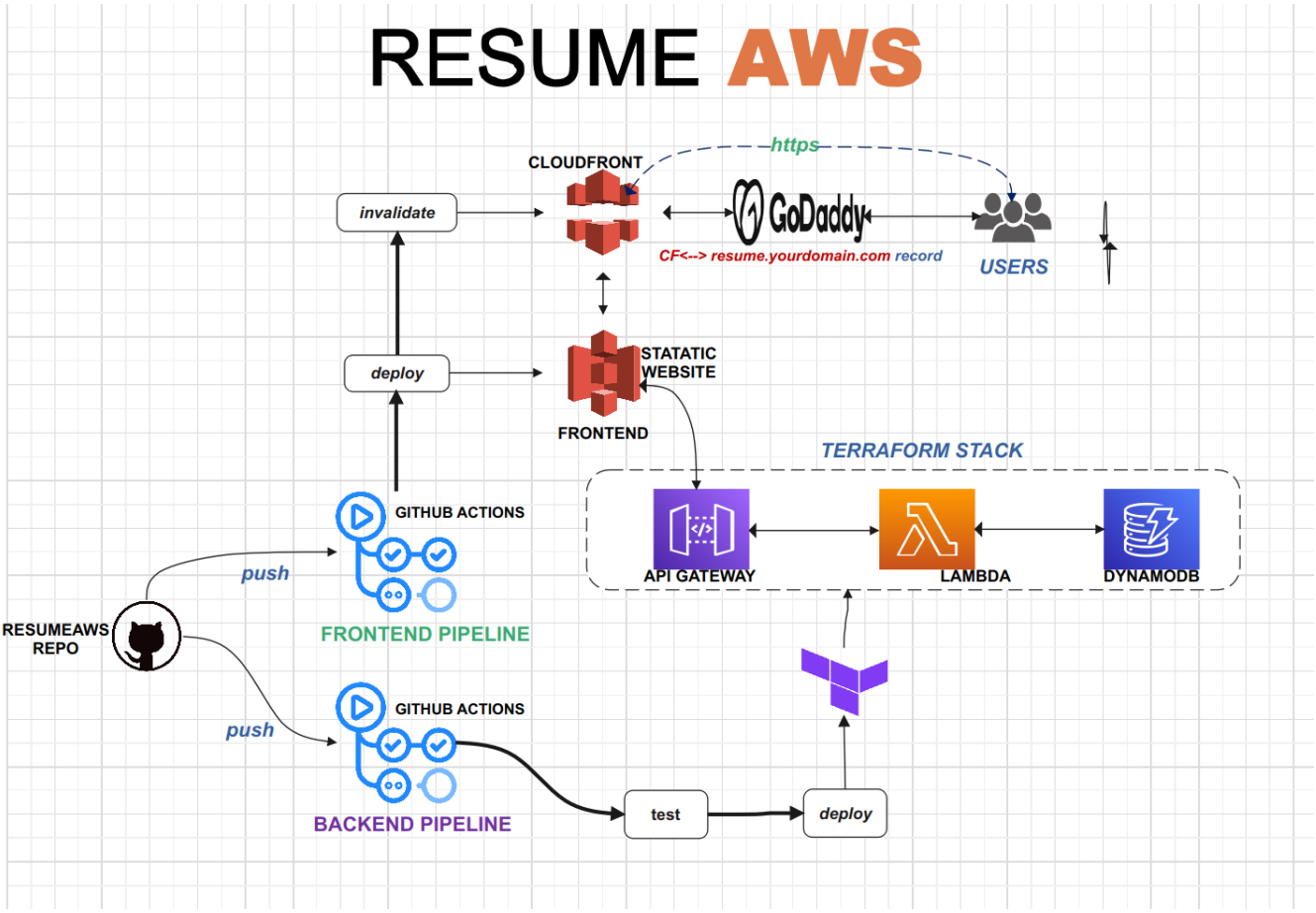
Project Title: **Resume Challenge on AWS**
Project Source: https://cloudresumechallenge.dev/docs/the-challenge/aws/
Project Repo: ***https://github.com/robudexIT/resume-aws***

Project Architecure:



Project Summary:

1. **Overview:** This project focuses on building a Simple WebApps project while applying DevOps principles throughout its Software Development Life Cycle (SDLC).
2. **Frontend Deployment:** Frontend development involves HTML, CSS, and JavaScript. The application will be deployed to an S3 static website with CloudFront for content delivery.
3. **Custom Domain Setup:** Custom domain configuration will be handled using GoDaddy.com for seamless access to the deployed application.
4. **Backend Development:** The backend architecture will utilize AWS Serverless services, employing **API Gateway, Lambda Functions, and DynamoDB**. Infrastructure provisioning will be managed using **Terraform.**

5. **Testing :** Testing will be conducted using **Pytest** because Lambda functions us Python
6. **Source Code Management:** **GitHub** will serve as the source code repository, facilitating collaborative development and version control.
7. **CI/CD Pipeline:** GitHub Actions will be employed to automate Continuous Integration and Continuous Deployment (CI/CD) processes.

Project Details:

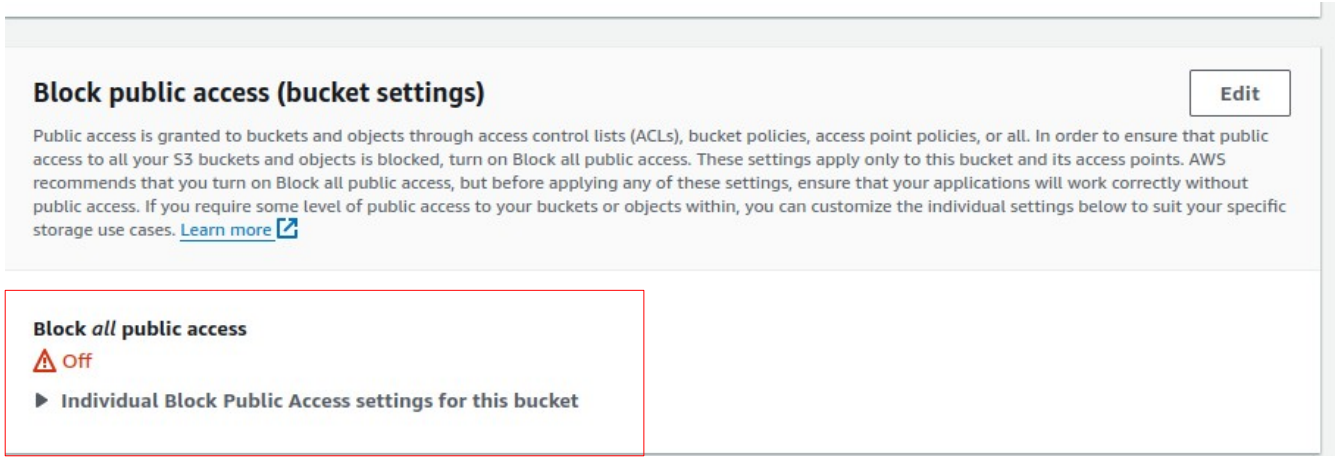
1. ***Below is my project structure. I've highlighted only the relevant files***

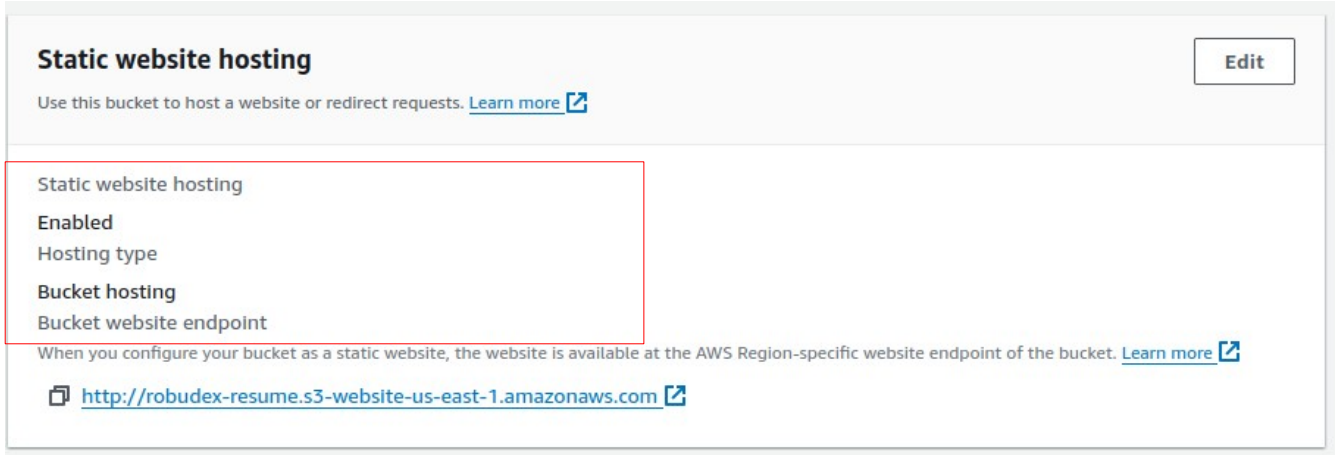
```
├── backend
│   ├── a.out
│   ├── backend_github_ctions.yaml
│   ├── hello.c
│   ├── __init__.py
│   ├── lambda_code
│   │   ├── index.py – this file will be deploy on our lambda function
│   │   └── __init__.py
│   ├── requirements.txt  - this file content the python package need on this project
│   ├── resume-ai.tf
│   ├── resumecf.yaml
│   ├── resume.yaml
│   ├── terraform
│   │   ├── resume.tf   - this file contain  our backend infrastructure
│   │   └── variable.tf  -
│   └── test_code.py – this file contain our python test code
└── frontend   - all files in this folder will be uploaded to our S3 bucket
    ├── index.html
    └── myresume_photos.jpg
```

My CICD pipelines are place in the hidden directory **.github** cd to it and type **tree** command as well

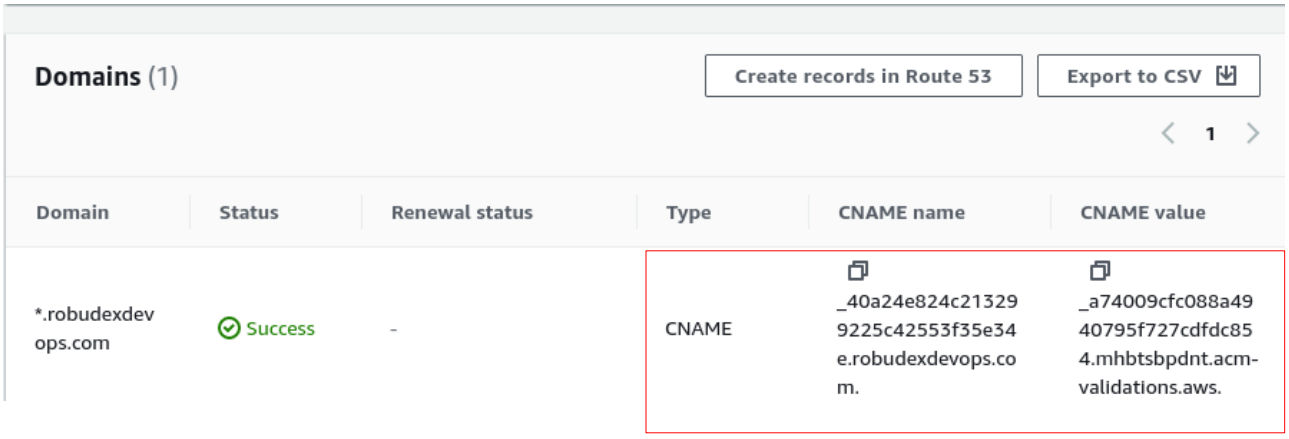
```
└── workflows
    ├── resume_backend_pipeline.yml  - this file is our code for backend pipeline
    └── resume_frontend_pipeline.yml  - this file is  our code for frontend pipeline
```

2. I Create S3 Bucket name **robudex-resume**, ***enable static website hosting*** *and turn the* ***Block all public access*** *off (this will serve as my frontend of my app)*





3. Because I plan to use a custom domain I need Certificate to make this happen. I request certificate to ***AWS Certificte Manager (ACM)***



I Add this CNAME records to my ***godaddy DNS Records*** *and wait for the status to Success.*

CNAME	_40a24e824c213299225c42553f35e34e	_a74009cfc088a4940795f727cdfdc854.mhbtsbpdnt.acm-validations.aws.	1 Hour	

*4.Next I Created **Cloudfront Distribution** with the S3 bucket that I created earlier as my origin. Because I want the Origin access can only be access by the Cloudfront I set **Origin access control settings (**which is by the way its recommended).*

- *For Alternate domain name (CNAME) I used my choosen domain name: **resumeaws.robudexdevops.com***
- *For Custom SSL certificate, I Choose the Certificate that I created*
- *I set also my Default root Object to **index.html***
- For the Viewer Setting, I set **Redirect HTTP to HTTPS**

Alternate domain name (CNAME) - *optional*

Add the custom domain names that you use in URLs for the files served by this distribution.

resumeaws.robudexdevops.com | Remove

Add item

To add a list of alternative domain names, use the bulk editor.

Custom SSL certificate - *optional*

Associate a certificate from AWS Certificate Manager. The certificate must be in the US East (N. Virginia) Region (us-east-1).

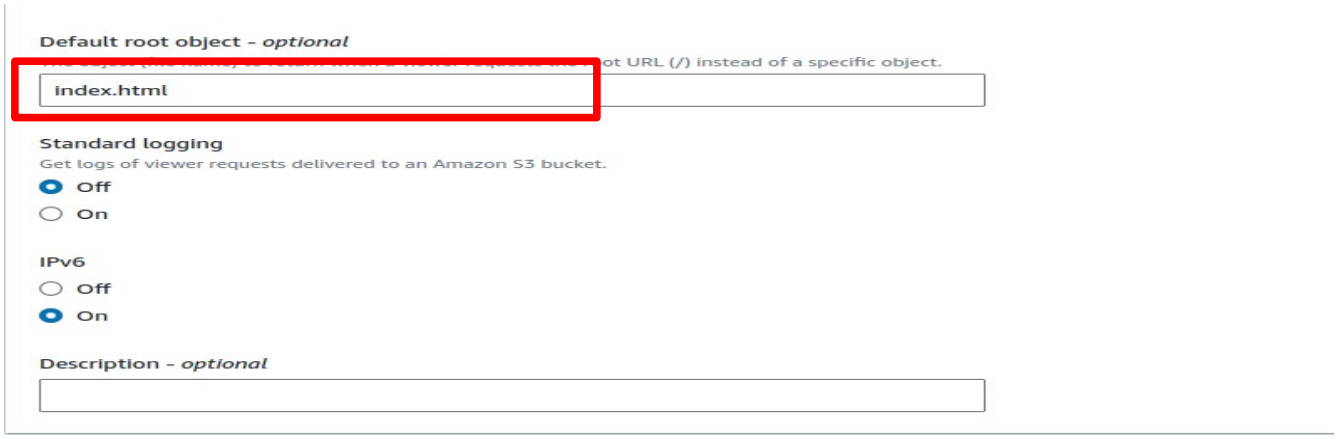
*.robudexdevops.com (ae470842-03a2-4662-9d3c-1accf0462c6c)

*.robudexdevops.com Request certificate

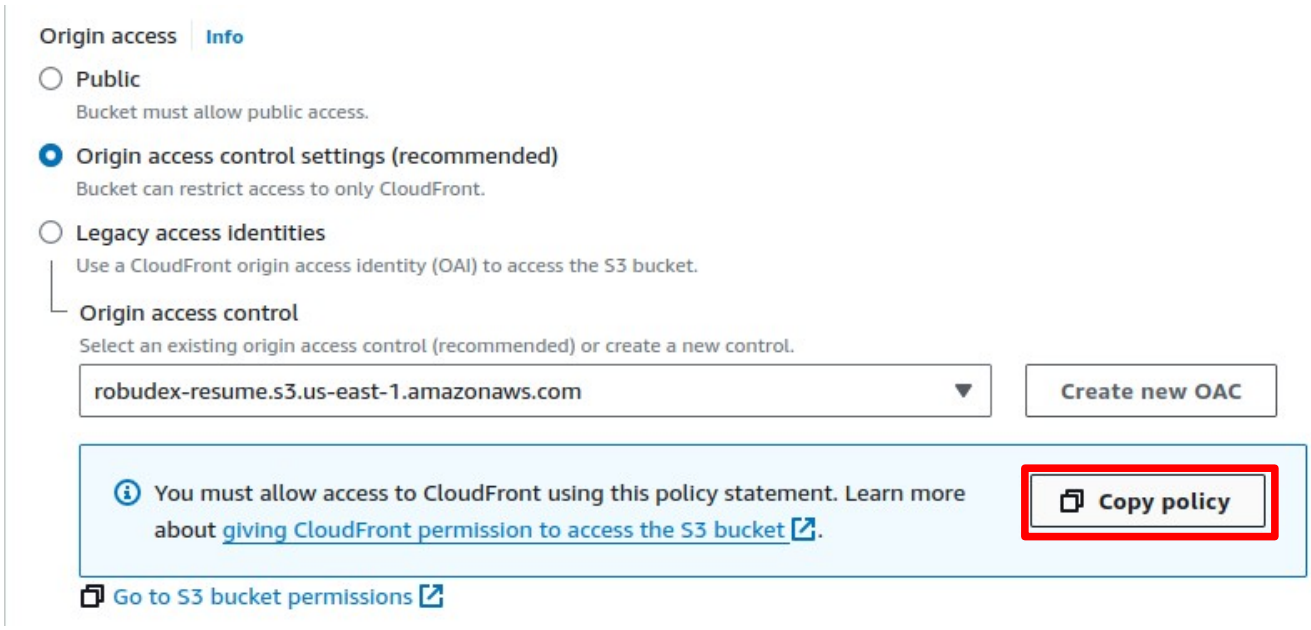
Legacy clients support - $600/month prorated charge applies. Most customers do not need this.

CloudFront allocates dedicated IP addresses at each CloudFront edge location to serve your content over HTTPS.

Enabled



5. **Once the CloudFront Distribution is created.**
   - *I Click my Distribution ID,*
   - *Goto Origins and click edit*
   - *On the origin Access I copy the policy and paste it ito the Bucket Policy of my bucket orgin*

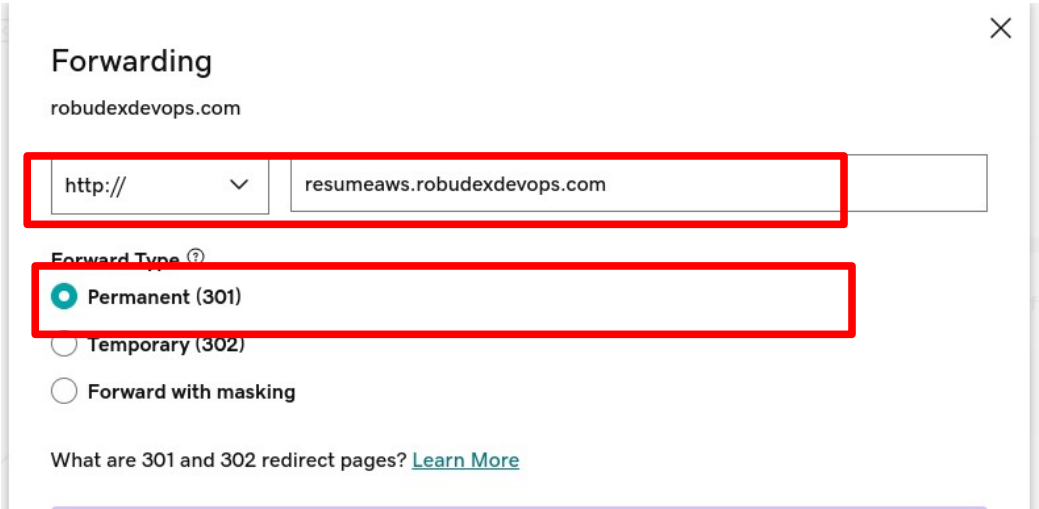


```
{
  "Version": "2008-10-17",
  "Id": "PolicyForCloudFrontPrivateContent",
  "Statement": [
    {
      "Sid": "AllowCloudFrontServicePrincipal",
      "Effect": "Allow",
      "Principal": {
        "Service": "cloudfront.amazonaws.com"
      },
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::robudex-resume/*",
      "Condition": {
        "StringEquals": {
          "AWS:SourceArn": "arn:aws:cloudfront::700392114790:distribution/E3PIHBMUV4MOOL"
        }
      }
    }
  ]
}
```

Copy

6. **Update my godaddy.com account to point resumeaws.robudexdevops.com to my Cloudfront Distribution**

- *Under My robudexdevops.com domain I created CNAME records resumeaws --> ${MY_CLOUDFRONT_DOMAIN}*
- *Setup Forwading http://resumeaws.robudexdevops.com --> redirect 301*



7. **Now that my frontend is set. Its time to build the backend**

A. *For my lambda sdffunction, it is located in* **backend/lambda_code/** it contains:

- **index.py –** the source code.
- **index.zip –** the package that will use by terraform for creating lambda service.

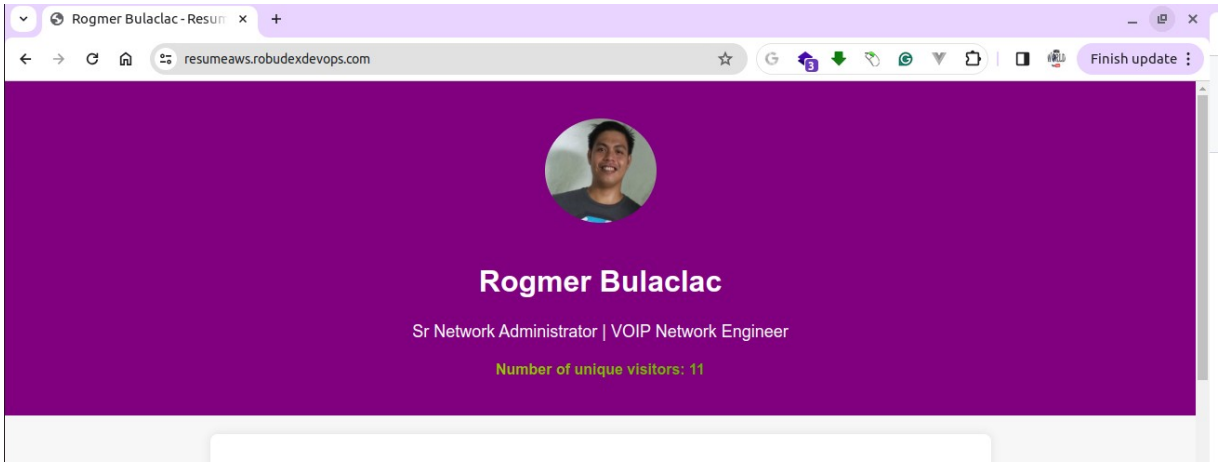
B. Build backend Infrastrucure using Terraform

- *I* ***cd*** *to my backend/terraform project folder (which contain all codes for my backend infrastructure)*
- ***resume.tf i****s the file where the aws services are declared to be created*
- *And run nesssary commands to build the infrastructure (****terraform init, terrafrom fmt, terraform validate, terraform plan and terraform apply****)*
- *After Terraform finishes creating the infrastructure, copy the newly created API Endpoint from the output. Append '****/resumeaws****' to the end of the URL.*
- *Open the frontend/index.html and udpate the API endpoint in the javascript section of the file.*
- *Upload the update index.html file to S3 bucket*

```
No changes. Your infrastructure matches the configuration.

Terraform has compared your real infrastructure against your configuration and found no differences, so no changes are needed.

Apply complete! Resources: 0 added, 0 changed, 0 destroyed.

Outputs:

api_gateway_url = "https://40jhq3k8u6.execute-api.us-east-1.amazonaws.com/staging"
```

```
<script>
    // Function to fetch and display visitor count
    async function fetchVisitorCount() {
        try {
            const response = await fetch('https://40jhq3k8u6.execute-api.us-east-1.amazonaws.com/staging/resumeaws');
            const data = await response.json();
            console.log(data)
            document.getElementById('visitorCount').innerHTML = `Number of unique visitors: ${data}`;
        } catch (error) {
            console.error('Error fetching visitor count:', error);
        }
    }

    // Call the function when the page is loaded
    window.onload = fetchVisitorCount;


</script>
```

8 **Time Test the App. I open my browser the paste resumeaws.robudexdevops.com**



And its working!! with ***https secure connections.*** *Great!! Its time to build the final piece - the most fun part-setting up the Github repository, uploading the code,and creating CI/CD pipelines*

9. **Enable my project repo to use git**

- *git init*
- *touch .gitignore*
- ***echo "**/.terraform/*" > .gitignore***
- *git add .*
- *git commit -m "First Comment"*
- *git checkout -b main*
- *git branch -d master*
-

Note: (I create branch called **main** *and delete my* **master** *branch because github now is using* ***main*** *branch as master branch)*

10. **Creating Github Repo**

- *On my Github I created new repository called* ***resume-aws,*** *copy this code*

  git remote add origin git@github.com:robudexIT/resume-aws.git
  git branch -M main
  git push -u origin main

11. **Add the origin to my project repo and push it to github**

- *git remote add origin* git@github.com*:robudexIT/resume-aws.git*
- *git push -u origin main.*

12. **Now that all are set. Its time to create CICD using github actions.**

- *On my project I created* ***.github/workflows*** *directory (mkdir* ***.github/workflows****) cd to it (cd* ***.github/workflows****) and created two files*
- ***resume_backend_pipeline.yml*** *-> pipeine for my backend*
- ***resume_frontend_pipeline.yml*** *-> pipeline for my frontend*

resume_backend_pipeline.yml

```yaml
name: resume-backend-pipeline

on:
  push:
    paths:
      - "backend/**"
    branches:
      - main

jobs:
  test-code:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout code
        uses: actions/checkout@v4

      - name: Set up AWS CLI
        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: us-east-1  # Replace with your AWS region

      - name: Set up Python
        uses: actions/setup-python@v5
        with:
            python-version: 3.9


      - name: Install dependencies
        run: |
          cd backend
          python -m pip install --upgrade pip
          pip install -r requirements.txt

      - name: Run test
        run:  |
          cd backend
          pytest
```

It will trigger only when push to main branch and on backend/ path**

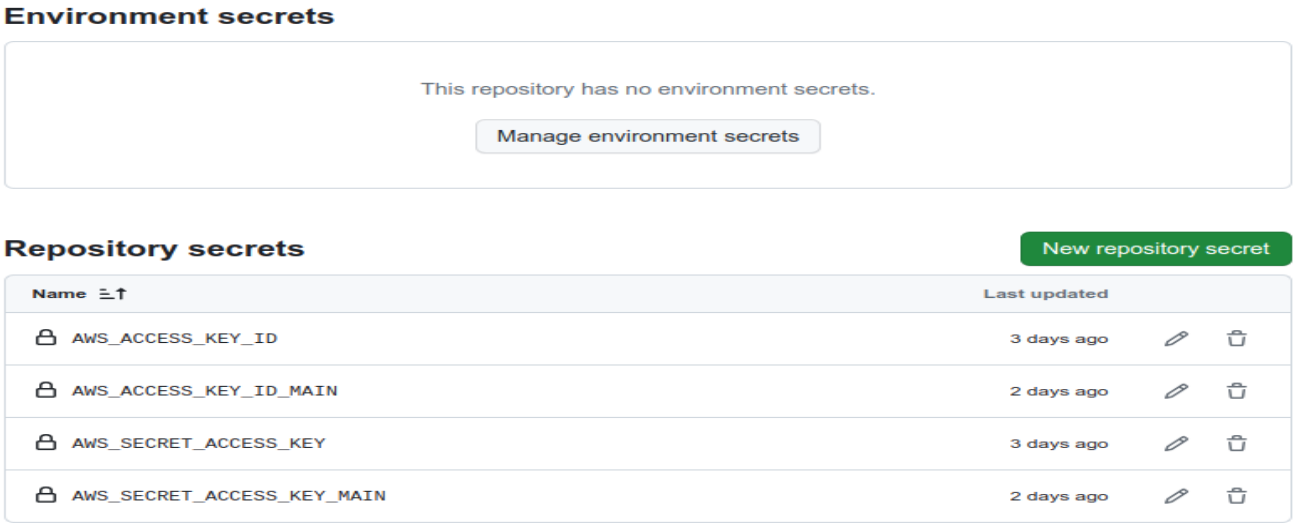
```yaml
deploy-terraform:
  runs-on: ubuntu-latest

  steps:
    - name: Checkout code
      uses: actions/checkout@v4

    - name: packaging code
      run: |
        cd backend/lambda_code
        timestamp=$(date "+%Y-%m-%d_%H-%M-%S")
        zip index-${timestamp}.zip index.py
        echo "lambda_code_file = \"index-${timestamp}.zip\"" > ../terraform/terraform.tfvars

    - name: Setup Terrform
      uses: hashicorp/setup-terraform@v3

    - name: Setup AWS CLI
      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: us-east-1  # Replace with your AWS region

    - name: Terraform Init
      id: init
      run: |
        cd backend/terraform
        terraform init

    - name: Terraform Format
      id: fmt
      run: |
        cd backend/terraform
        terraform fmt
    - name: Terraform Validate
      id: Validate
      run: |
        cd backend/terraform
        terraform validate

    - name: Terraform Plan
      id: plan
      run: |
        cd backend/terraform
        terraform plan -var-file="terraform.tfvars"
    - name: Terraform Apply
      id: apply
      run: |
        cd backend/terraform
        terraform apply -var-file="terraform.tfvars" -auto-approve=true
```

resume_frontend_pipeline.yml

```yaml
name: resume-backend-pipeline

on:
  push:
    paths:
      - "frontend/**"
    branches:
      - main

jobs:
  copy-code:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout code
        uses: actions/checkout@v4

      - name: Set up AWS CLI
        uses: aws-actions/configure-aws-credentials@v1
        with:
          aws-access-key-id: ${{ secrets.AWS_ACCESS_KEY_ID_MAIN }}
          aws-secret-access-key: ${{ secrets.AWS_SECRET_ACCESS_KEY_MAIN }}
          aws-region: us-east-1  # Replace with your AWS region

      - name: Deploy to S3
        run: |
          cd frontend
          aws s3 sync . s3://robudex-resume --delete
      - name: Invalidate Cloudfront Cache
        run: |
          aws cloudfront create-invalidation --distribution-id E3PIHBMUV4MOOL --paths "/*"
```

It will trigger only when push to main branch and on frontend/ path**

Note: Because it is very risky and not recommended to put any credentials on the pipelines, I use the secret.



13. **Time to Test the pipelines**

- *For Backend pipelines - this is my current lambda code*

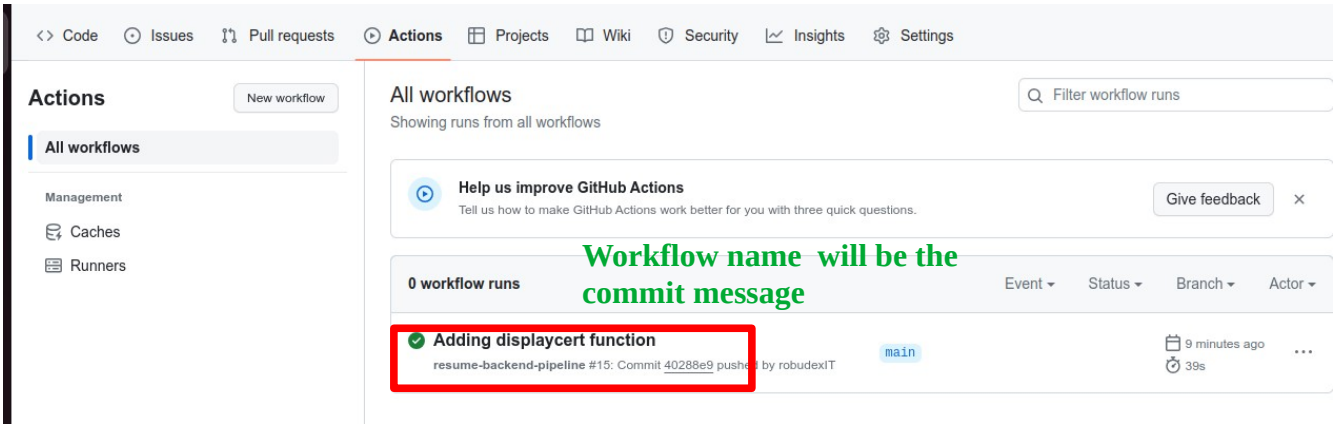
Current state

```python
import json
import boto3

dynamodb = boto3.resource('dynamodb')
table_name = 'resume_visitor'
table = dynamodb.Table(table_name)



def displayname():
    print("Author:Rogmer Delacruz Bulaclac")
    print("Position: Devops Engineer")

def lambda_handler(event, context):

    try:
        client_ip = event['requestContext']['identity']['sourceIp']
        # Check if the IP address is already in the database
        existing_visitor = table.get_item(
            Key={'ip': client_ip}
        )

        if 'Item' not in existing_visitor:
            # If the IP address is not in the database, add a new entry
            table.put_item(
                Item={'ip': client_ip}
            )

        # Get the count of unique visitors
        response = table.scan(Select='COUNT')


        return  {
            'statusCode': 200,
             'headers': {
                "Access-Control-Allow-Headers" : "Content-Type",
                "Access-Control-Allow-Origin": "*",
                "Access-Control-Allow-Methods": "OPTIONS,POST,GET"
            },
            'body': f'{response["Count"]}'
         }
```

- *Maybe I will add another function callled **displaycert** and push it*

```
def displaycert():
        print("AWS Developer Associate")
        print ("AWS Solution Architect Associate")
```

```
        lambda_code/index.zip
        ../resumeaws.odt

no changes added to commit (use "git add" and/or "git commit -a")
(base) robudex@robudex-Dell-System-Vostro-3360:~/SBTPHPROJECTS/resume-aws/backend$ git add .
(base) robudex@robudex-Dell-System-Vostro-3360:~/SBTPHPROJECTS/resume-aws/backend$ git commit -m "Adding displaycert function"
[main 40288e9] Adding displaycert function
 5 files changed, 286 insertions(+), 24 deletions(-)
 create mode 100644 backend/lambda_code/index.zip
(base) robudex@robudex-Dell-System-Vostro-3360:~/SBTPHPROJECTS/resume-aws/backend$ git push
Enumerating objects: 18, done.
Counting objects: 100% (18/18), done.
Delta compression using up to 4 threads
Compressing objects: 100% (10/10), done.
Writing objects: 100% (10/10), 3.36 KiB | 1.68 MiB/s, done.
Total 10 (delta 5), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (5/5), completed with 4 local objects.
To github.com:robudexIT/resume-aws.git
   210fc26..40288e9  main -> main
```

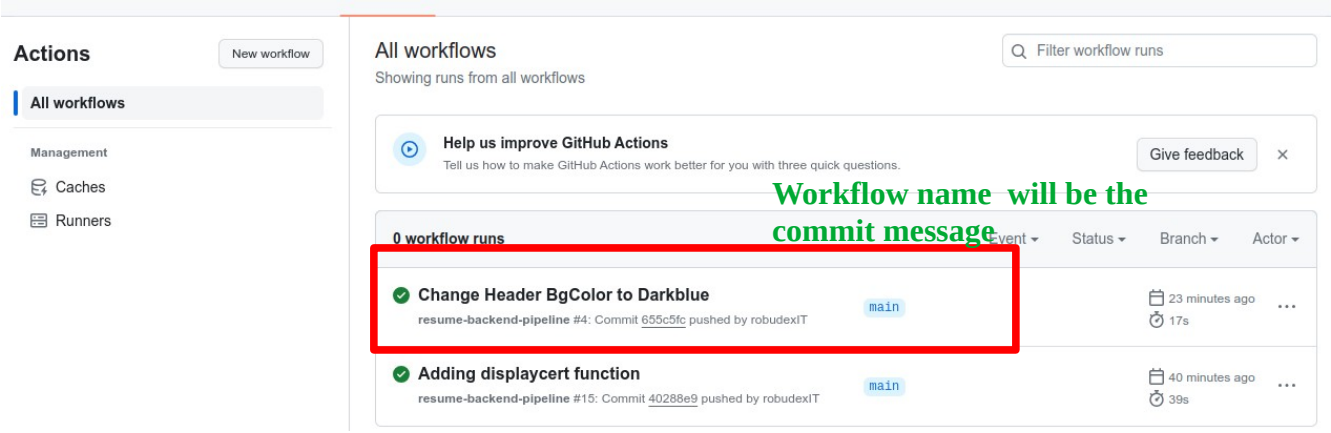


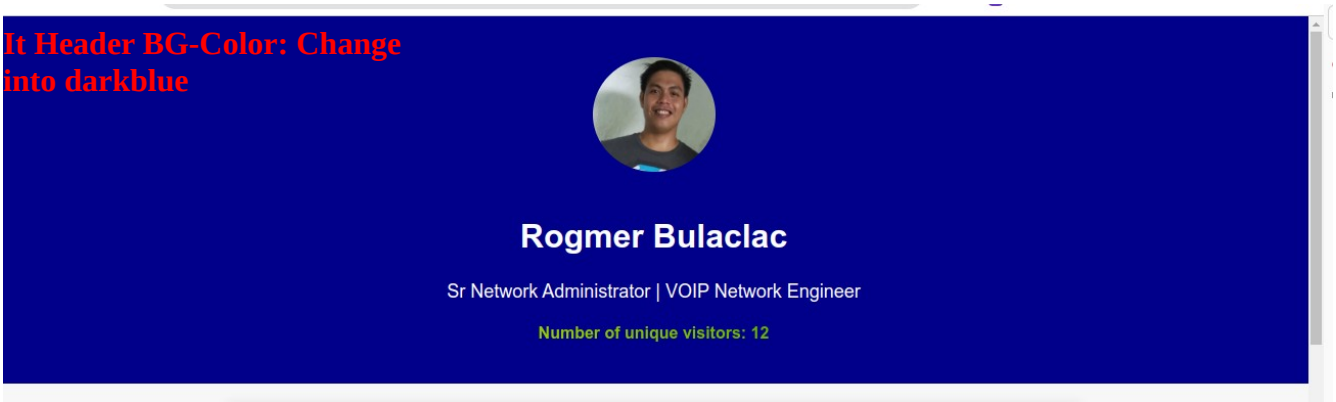
```python
import json
import boto3

dynamodb = boto3.resource('dynamodb')
table_name = 'resume_visitor'
table = dynamodb.Table(table_name)



def displayname():
    print("Author:Rogmer Delacruz Bulaclac")
    print("Position: Devops Engineer")

def displaycert():
    print("AWS Developer Associate")
    print ("AWS Solution Architect Associate")

def lambda_handler(event, context):

    try:
        client_ip = event['requestContext']['identity']['sourceIp']
        # Check if the IP address is already in the database
        existing_visitor = table.get_item(
            Key={'ip': client_ip}
        )

        if 'Item' not in existing_visitor:
            # If the IP address is not in the database, add a new entry
            table.put_item(
                Item={'ip': client_ip}
            )

        # Get the count of unique visitors
        response = table.scan(Select='COUNT')


        return  {
            'statusCode': 200,
             'headers': {
                "Access-Control-Allow-Headers" : "Content-Type",
                "Access-Control-Allow-Origin": "*",
                "Access-Control-Allow-Methods": "OPTIONS,POST,GET"
            },
```

displaycert() function is added

- For Frontend, the current header bg-color is **dark purple**, let change it to **dark blue** and push it

```
(base) robudex@robudex-Dell-System-Vostro-3360:~/SBTPHPROJECTS/resume-aws$ git add .
(base) robudex@robudex-Dell-System-Vostro-3360:~/SBTPHPROJECTS/resume-aws$ git commit -m "Change Header BgColor to Darkblue"
[main 655c5fc] Change Header BgColor to Darkblue
 3 files changed, 3 insertions(+), 3 deletions(-)
 create mode 100644 .~lock.resumeaws.odt#
 create mode 100644 resumeaws.odt
(base) robudex@robudex-Dell-System-Vostro-3360:~/SBTPHPROJECTS/resume-aws$ git push
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 4 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 964.50 KiB | 3.05 MiB/s, done.
Total 6 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:robudexIT/resume-aws.git
   40288e9..655c5fc  main -> main
```


Actions
New workflow
All workflows
Management
Caches
Runners
All workflows
Showing runs from all workflows
Filter workflow runs
Help us improve GitHub Actions
Tell us how to make GitHub Actions work better for you with three quick questions.
Give feedback
Workflow name will be the commit message
0 workflow runs
Event
Status
Branch
Actor
Change Header BgColor to Darkblue
resume-backend-pipeline #4: Commit 655c5fc pushed by robudexIT
main
23 minutes ago
17s
Adding displaycert function
resume-backend-pipeline #15: Commit 40288e9 pushed by robudexIT
main
40 minutes ago
39s


It Header BG-Color: Change into darkblue
Rogmer Bulaclac
Sr Network Administrator | VOIP Network Engineer
Number of unique visitors: 12