

# **ANASIEZE IKENNA – CLOUD & AI SOULTIONS ENGINEER**

## **Project: CI CD Pipeline with GitHub Actions and AWS**

### **Overview**

I implemented a production grade CI/CD pipeline using GitHub Actions and AWS. The pipeline automates build, test, security scanning, and container publishing for a Java application across multiple environments.

### **Problem Statement**

Engineering teams needed a CI/CD solution that could:

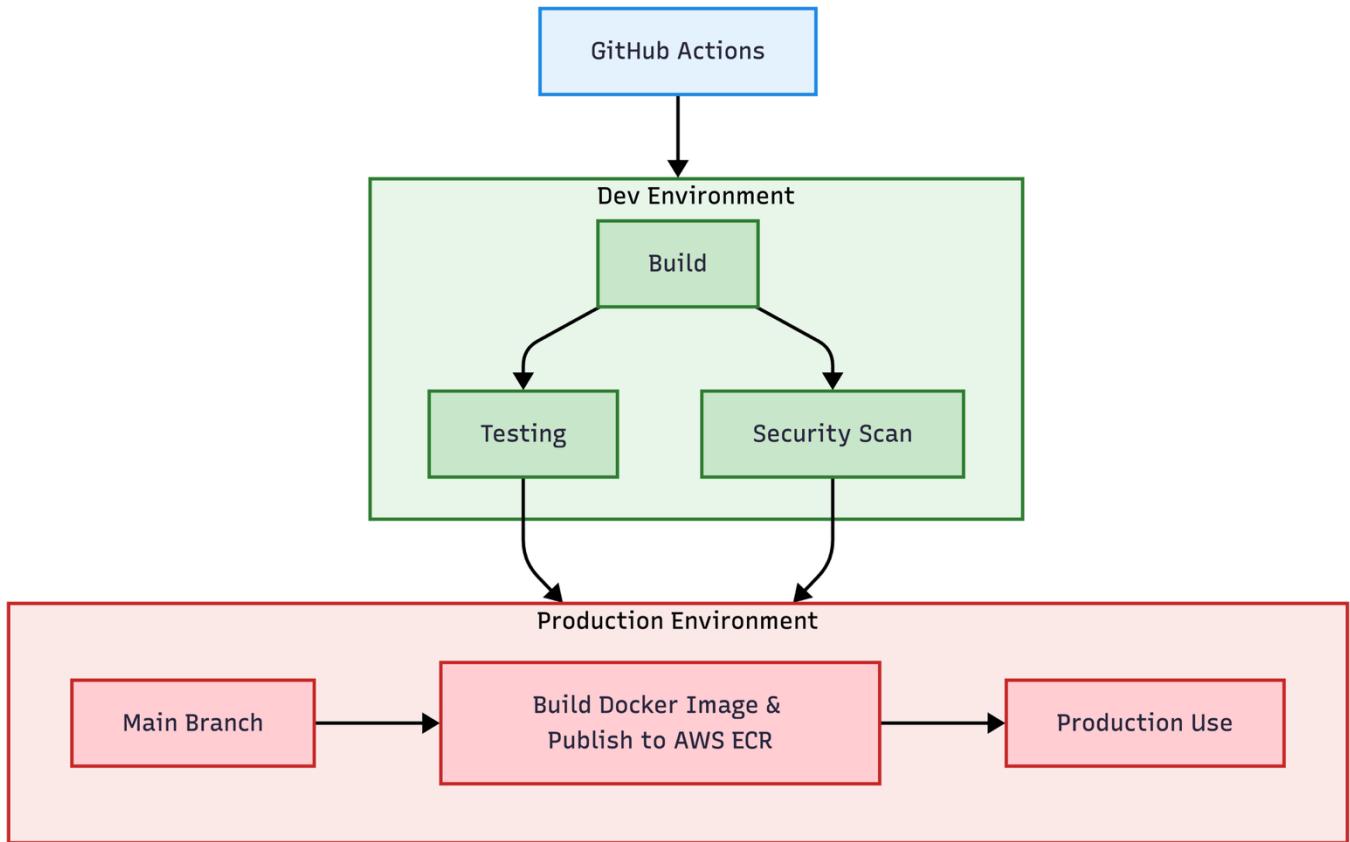
- + Support multiple environments without leaking secrets
- + Prevent accidental production deployments
- + Enforce testing and security checks before release
- + Provide repeatable and auditable deployments
- + Integrate securely with AWS container infrastructure

**Goal:** Responsible for designing and implementing an end to end CI/CD workflow that supports **multiple environments** using GitHub Actions.

### **Tech Stack**

- + GitHub Actions – CI/CD orchestration
- + Maven – Java build and packaging
- + Docker – Application containerization
- + Trivy – Security and vulnerability scanning
- + Amazon ECR – Container image registry
- + AWS IAM – Environment scoped access control

## Architecture Overview



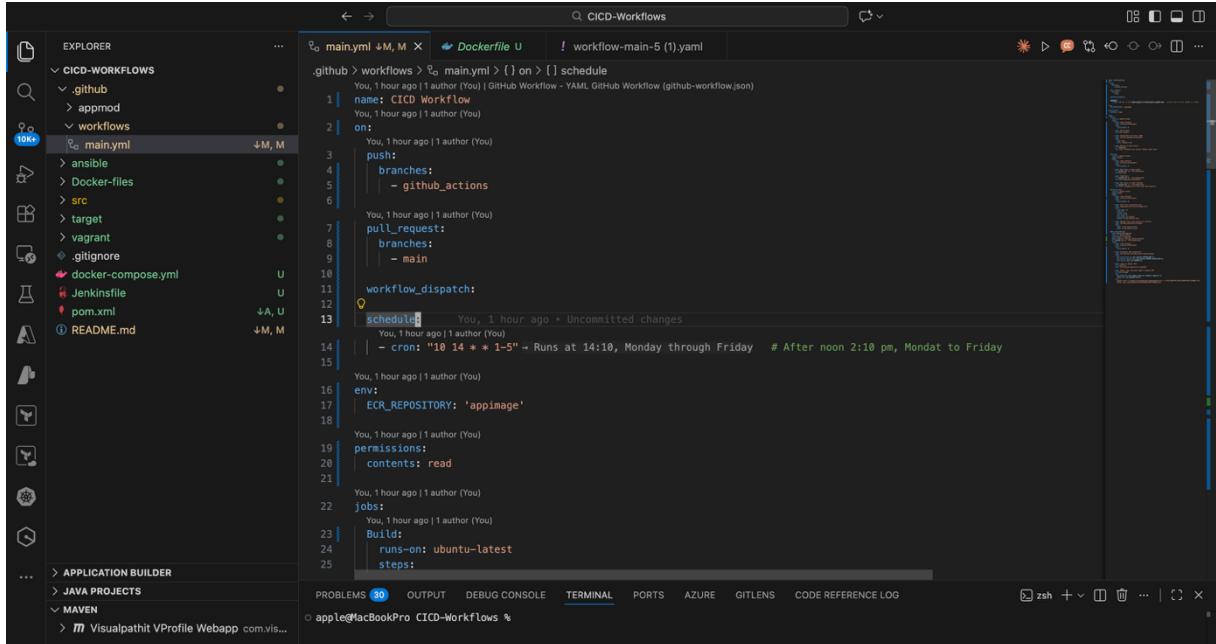
### Flow of Pipeline

- + Code changes are pushed to feature branches or submitted via pull requests
- + CI jobs run automatically to build the application and generate artifacts
- + Security scanning runs in parallel with build validation
- + Tests and static analysis run when changes reach the main branch
- + Production release is gated and triggered only from main
- + Docker images are built and published to Amazon ECR using production scoped credentials

This design ensures production remains protected while allowing fast iteration in lower environments.

## Deployment Steps

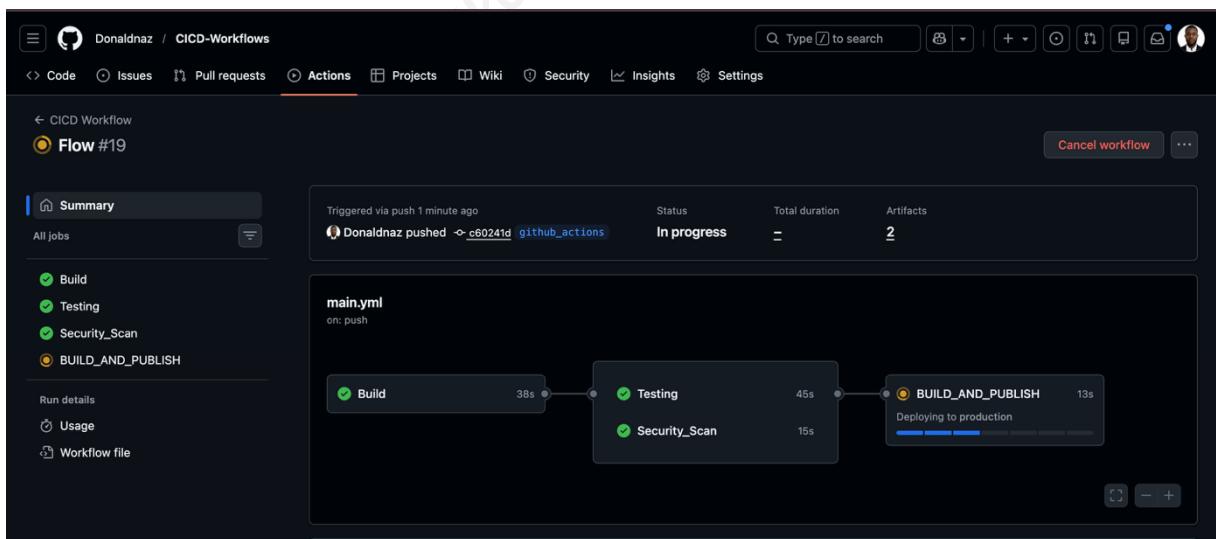
- + Created a GitHub Actions workflow with job dependencies



The screenshot shows a code editor interface with a dark theme. On the left is the Explorer sidebar, which lists several files and folders under a 'CICD-WORKFLOWS' directory, including 'main.yml', 'Dockerfile', and 'workflow-main-5 (1).yaml'. The main editor area displays the content of 'main.yml'. The code defines a GitHub Action workflow with various triggers, branches, and jobs. It includes sections for 'push', 'pull\_request', and 'workflow\_dispatch'. A 'schedule' section is present with a cron expression: '0 10 \* \* \* -> Runs at 14:10, Monday through Friday # After noon 2:10 pm, Monday to Friday'. The 'env' section specifies an ECR repository named 'appimage'. The 'permissions' section grants 'read' access. The 'jobs' section contains a single 'Build' job that runs on 'ubuntu-latest' and has steps defined.

```
name: CICD Workflow
on:
  push:
    branches:
      - github_actions
  pull_request:
    branches:
      - main
  workflow_dispatch:
  schedule:
    You, 1 hour ago + Uncommitted changes
    You, 1 hour ago | 1 author (You)
    - cron: "0 10 * * * -> Runs at 14:10, Monday through Friday # After noon 2:10 pm, Monday to Friday"
  env:
    ECR_REPOSITORY: 'appimage'
  permissions:
    contents: read
  jobs:
    Build:
      runs-on: ubuntu-latest
      steps:
```

- + Configured build, test, and security scan stages



- + Protected production deployment using a GitHub environment

The screenshot shows two sections of a GitHub environment configuration:

### Environment secrets

Secrets are encrypted environment variables. They are accessible only by GitHub Actions in the context of this environment by using the [secret context](#).

Name	Last updated
ACCESS_KEY	19 hours ago
SECRET_ACCESS_KEY	19 hours ago

### Environment variables

Variables are used for non-sensitive configuration data. They are accessible only by GitHub Actions in the context of this environment by using the [variable context](#).

Name	Value	Last updated
REGION	us-east-2	19 hours ago

- + Built and published Docker images to Amazon ECR from main

The screenshot shows the AWS ECR console under the Private registry section, specifically for the repository "appimage".

**Images (3) Info**

Image tags	Type	Created at	Image size	Image digest	Last pulled at
a2fa942eeb544b92d985fc5f8f0473acd9efacba	Image	02 February 2026, 23:59:49 (UTC-03:5)	312.00	sha256:0b9602312...	-
-	Image	02 February 2026, 23:51:02 (UTC-03:5)	312.00	sha256:d029c18c5...	-
c60241d540eb8b5cd41a385fe530153ae2de7fce	Image	02 February 2026, 23:42:14 (UTC-03:5)	312.00	sha256:3b3462145...	-

## Outcome

- A fully operational multi environment CI/CD pipeline
- Safe promotion of code from development to production
- Secure and repeatable Docker image publishing to ECR

## Key Takeaway

CI/CD is not about YAML files. It is about **control, safety, and repeatability at scale.**