**GITHUB CI/CD PLAY BOOK**

**Contents**

[**INTRODUCTION:** 1](#_Toc134526341)

[**GITHUB ACTIONS:** 2](#_Toc134526342)

[**TRIGGERS:** 3](#_Toc134526343)

[**JOBS:** 5](#_Toc134526344)

[**STEPS:** 5](#_Toc134526345)

[**CODE CHECKOUT** 5](#_Toc134526346)

[**GET THE DEPENDENCIES INSTALL** 5](#_Toc134526347)

[**SAST SCAN** 6](#_Toc134526348)

[**BUILD USING DEPENDENCIES** 7](#_Toc134526349)

[**DAST SCAN** 7](#_Toc134526350)

[**UPLOAD ARTIFACT** 8](#_Toc134526351)

[**DOWNLOAD ARTIFACT** 8](#_Toc134526352)

[**DEPLOYMENT** 8](#_Toc134526353)

[**SAMPLE YAML FOR INDENTATION PURPOSE:** 9](#_Toc134526354)

## **INTRODUCTION:**

GitHub Actions is a powerful automation platform provided by GitHub that allows you to automate your software development workflows. It provides a way to automate your software development workflows, including building, testing, and deploying your code.

With GitHub Actions, you can:

* Define custom workflows as code that can be versioned and reviewed like any other code in your repository.
* Use pre-built actions or write your own custom actions to encapsulate reusable pieces of code that you can use across multiple workflows.
* Run workflows on a range of different environments, including Linux, macOS, and Windows.
* Trigger workflows based on a variety of events, such as a pull request, a code push, or a schedule.
* Integrate with other services, tools, and platforms in your development pipeline.

Some of the things that you can do with GitHub Actions include:

**Continuous Integration (CI):** Automatically build and test your code when you push changes to a repository.

**Continuous Delivery (CD):** Automatically deploy your code to production or other environments when your tests pass, and your code is ready.

**Code quality checks**: Automatically run linters, style checkers, and other code quality tools to ensure that your code meets your team's standards.

**Security scans:** Automatically scan your code for security vulnerabilities and report issues to your team.

**Custom workflows:** Define custom workflows that are tailored to your team's specific needs.

Overall, GitHub Actions provides a powerful and flexible way to automate your software development workflows, enabling you to streamline your development process and focus on building high-quality software.

## **GITHUB ACTIONS:**

**Diagram

Description automatically generated**

Ideal flow of project CI/CD in GitHub Actions

* **Events:** Events are triggers that start a workflow. They can be things like pushing code to a repository, creating, or updating a pull request, or a scheduled event.
* **Workflows:** Workflows define the series of jobs that will be run in response to an event. Workflows are defined using YAML files that are stored in your repository.
* **Jobs:** Jobs are the individual units of work that make up a workflow. Jobs can run in parallel or sequentially and can be run on different machines or environments.
* **Steps:** Steps are the individual tasks that make up a job. Steps can include running a command, executing a script, or calling an external service. Steps can be pre-built actions provided by GitHub or custom actions that you create yourself.
* **Actions:** Actions are reusable pieces of code that encapsulate a set of steps. Actions can be shared and reused across different workflows, making it easy to standardize your development process.
* **Environment variables:** Environment variables are used to pass information between steps and jobs in a workflow. They can be set globally or for a specific job.
* **Artifacts:** Artifacts are files that are created by a job and can be used by subsequent jobs in the same workflow. Artifacts can be used to share data between jobs or to provide build outputs.
* **Secrets:** Secrets are encrypted values that can be used in your workflows. Secrets are stored securely and can be used to store sensitive information like API keys or passwords.

## **TRIGGERS:**

Here we declare the triggers for how the workflow should start. These are the different types of triggers we can use.

#if the workflow has to be triggered manually

on : workflow\_dispatch

#if the workflow has to be triggered manually in a specific environment

on:

workflow\_dispatch:

inputs:

environment:

description: 'Environment to run against'

type: environment

required: true

#if any push happened to the repository this workflow will be triggered

on: push

#if any push happened toa specific branch of the repository this workflow will be triggered

on:

push:

branches:

[<branch\_name>]

#if any pull request happened to the repository this workflow will be triggered

on: pull\_request

#if any push happened toa specific branch of the repository this workflow will be triggered

on:

pull\_request:

branches:

[<branch\_name>]

## **JOBS:**

#This is the sample code snippet for to start the job and on which running machine do we need to do the CI/CD runs on

jobs:

build:

runs-on: <RUNNING\_ MACHINE>

steps:

## **STEPS:**

These are the individual tasks that make up a job. Steps can include running a command, executing a script, or calling an external service. Steps can be pre-built.

### **CODE CHECKOUT**

#Use this action for checking out the code   
- name: checkout the code  
 uses actions/checkout@v3

### **GET THE DEPENDENCIES INSTALL**

#setting up the node.js environment for the project

- name: Setup Node.js environment

uses: actions/setup-node@v3.6.0

with:

node-version: <NODE\_VERSION>  
  
#setting up the maven environment for the project

- name: Set up JDK

uses: actions/setup-java@v2

with:

java-version: <JAVA\_VERSION>

#setting up the dotnet environment for the project  
 - name: Setup .NET Core

uses: actions/setup-dotnet@v1

with:

dotnet-version: <DOTNET\_VERSION >  
  
#setting up the python environment for the project  
 - name: Set up Python

uses: actions/setup-python@v4.6.0

with:

python-version: <PYTHON\_VERSION>

### **SAST SCAN**

#setting up the SonarQube for the project for linux runner

- name: SonarQube Scan   
 uses: sonarsource/sonarqube-scan-action@ v1.9.1

env:

SONAR\_TOKEN: ${{ secrets.SONAR\_TOKEN }}

SONAR\_HOST\_URL: ${{ secrets.SONAR\_HOST\_URL }}

#setting up the SonarQube for the project for windows runner

- name: Download SonarScanner

run: |

Invoke-WebRequest -Uri 'https://binaries.sonarsource.com/Distribution/sonar-scanner-cli/sonar-scanner-cli-4.6.2.2472.zip' -OutFile 'sonar-scanner.zip'

Expand-Archive -Path 'sonar-scanner.zip' -DestinationPath 'sonar-scanner'

Remove-Item 'sonar-scanner.zip'

- name: Set up SonarQube environment variables

run: |

$env:SONAR\_SCANNER\_HOME = "$Env:USERPROFILE/sonar-scanner/sonar-scanner-4.6.2.2472"

$env:PATH += ";$env:SONAR\_SCANNER\_HOME/bin"

- name: Run SonarQube analysis

shell: pwsh

run: |

$scannerPath = 'sonar-scanner\sonar-scanner-4.6.2.2472\bin\sonar-scanner.bat'

& cmd /c "$scannerPath" -D"sonar.projectBaseDir=$PWD" -D"sonar.config.path=./sonar-project.properties"

### **BUILD USING DEPENDENCIES**

#run the commands that need to build the project

V

### **DAST SCAN**

#setting up the OWASP zap scan for the project  
 - name: ZAP Scan

uses: zaproxy/action-full-scan@v0.4.0

with:

token: ${{ secrets.GITHUB\_TOKEN }}

docker\_name: <DOCKER\_NAME>

target: <TARGET\_URL>

rules\_file\_name: <PATH>

cmd\_options: '-a'

### **UPLOAD ARTIFACT**

#uploading the artifact that was build

- name: Upload artifact for deployment job

uses: actions/upload-artifact@v3

with:

name: <NAME\_OF \_THE\_ARTIFACT>

path: <PATH\_OF\_THE\_ARTIFACT>

### **DOWNLOAD ARTIFACT**

#download the uploaded artifact

- name: Download artifact from build job

uses: actions/download-artifact@v3

with:

name: <ARTIFACT\_NAME>

### **DEPLOYMENT**

# Deploy to Azure App Service

- name: Azure login

uses: azure/login@v1

with:

creds: ${{ secrets.AZURE\_CREDENTIALS }}

- name: Deploy to Azure App Service

uses: azure/webapps-deploy@v2

with:

#specify the name of the App that it was creted in the azure app services

app-name: ${{ secrets.PROD\_APP\_NAME }}

#specify the name of the deployment slot in which the deployment has to be done

slot-name: ${{ env.DEPLOYMENT\_SLOT\_NAME }}

#specify the package i.e. the artifact path

package: ${{ github.workspace }}/app-${{ github.run\_number }}.zip

#specify the publish profile which was created in the azure web app services

publish-profile: ${{ secrets.PROD\_PUBLISH\_PROFILE }  
  
  
  
#Deployment in any virtual machine  
deploy:

#for this we need to add the VM as the self-hosted runner in that repository settings

runs-on: [self-hosted]

needs: build

steps:

- name: Download artifact from build job

uses: actions/download-artifact@v3

with:

name: <ARTIFACT\_NAME>  
 path: <PATH\_TO\_DOWNLOAD>

#deployment in app engine

- name: 'Set up Cloud SDK'

uses: google-github-actions/setup-gcloud@v1

with:

version: '>= 363.0.0'

- name: 'Authenticate to Google Cloud'

uses: 'google-github-actions/auth@v1'

with:

credentials\_json: ${{ secrets.GOOGLE\_APPLICATION\_CREDENTIALS }}

- name: print the credentials

run: echo "${{ secrets.GOOGLE\_APPLICATION\_CREDENTIALS }}"

- name: Deploy to Google App Engine

run: |

gcloud app deploy

- name: get-the-app-engine-link

run: |

gcloud app browse -s capitalization-ui

#deployment in azure app function

- name: Azure login

uses: azure/login@v1

with:

creds: ${{ secrets.AZURE\_CREDENTIALS }}

- name: Set up Azure Functions Core Tools

uses: azure/functions-action@v1

with:

app-name: bireport-dev-func1

package: ./publish\_output

publish-profile: ${{ secrets.AZURE\_FUNCTIONAPP\_PUBLISH\_PROFILE }}

#deployment in azure Kubernetes

- name: Azure Login

uses: azure/login@v1

with:

creds: ${{ secrets.AZURE\_CREDENTIALS }}

- name: Fetch Kubeconfig

run: |

az aks get-credentials --resource-group <resource\_group\_name> --name <cluster\_name>

mkdir -p ${{ github.workspace }}/.kube

cp ~/.kube/config ${{ github.workspace }}/.kube/config

- name: Set up Kubernetes CLI

uses: azure/setup-kubectl@v1

with:

kubeconfig: ${{ github.workspace }}/.kube

- name: Apply Kubernetes configuration

run: |

kubectl apply -f ${{ github.workspace }}/<path\_to\_Kubernetes\_yaml\_file>

## **SAMPLE YAML FOR INDENTATION PURPOSE:**

name: CI

#trigger the workflow

on:

push

jobs:

build:

runs-on: ubuntu-latest

steps:

- name: Checkout

uses: actions/checkout@v3.3.0

#setting up the node.js environmentfor the project

- name: Setup Node.js environment

uses: actions/setup-node@v3.6.0

with:

node-version: <NODE\_VERSION>

#giving the commands for the npm to start and run the test

- name: Install NPM dependencies

run: <DEPENDENCY\_COMMANDS>

- name: run the test

run: <RUN\_TEST\_COMMANDS>

- name: Upload a Build Artifact

uses: actions/upload-artifact@v3.1.2

with:

name: <DESTINATION\_FOLDER\_NAME>

#specify the path from where that the artifact has to be upload to the github server.

path: <ARTIFACT\_SOURCE\_PATH>