GitHub Actions Tutorial - Basic Concepts and CI/CD Pipeline with Docker

Introduction

This document provides a comprehensive overview of the **GitHub Actions Tutorial - Basic Concepts and CI/CD Pipeline with Docker** project. It covers the fundamental concepts of GitHub Actions, developer workflows, CI/CD pipelines, and Docker integration. The document also includes placeholders for screenshots to illustrate various steps in the process.

---

1. What is GitHub Actions?

GitHub Actions is an automation tool that helps developers streamline their workflows directly within GitHub repositories. It allows for continuous integration and continuous deployment (CI/CD) of applications, automating testing, building, and deployment tasks.

---

2. Developer Workflows and Use Cases

Workflows in GitHub Actions define how code is built, tested, and deployed. They trigger actions based on repository events like **push**, **pull request**, or scheduled jobs.

Use cases include:

- Automating code testing
- Building applications
- Deploying to cloud environments

---

3. Basic Concepts of GitHub Actions

GitHub Actions consist of the following:

- **Workflows:** Automation processes defined in YAML files
- **Events:** Triggers that start workflows
- **Jobs:** Individual steps in a workflow
- **Actions:** Predefined or custom reusable units of work

---

4. GitHub Actions CI/CD

CI/CD workflows in GitHub Actions automate the process of testing and deploying code changes, ensuring faster development cycles.

---

5. Why Another CI/CD Tool?

GitHub Actions simplifies CI/CD integration by providing:

- Seamless GitHub repository integration
- Native workflow automation
- Cost-effective and scalable solutions

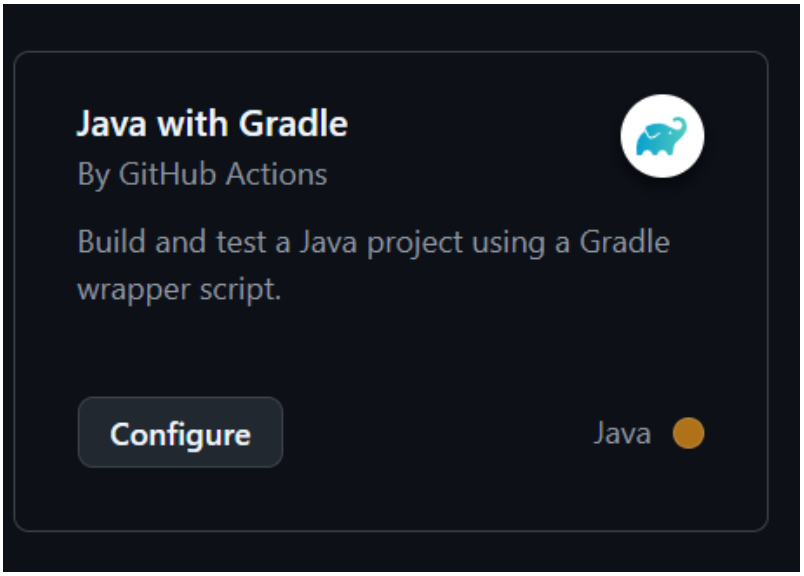
---

6. Demo: Creating a CI Workflow

A hands-on demonstration of setting up a **CI workflow** for a Java project using **Gradle** and **Docker**.

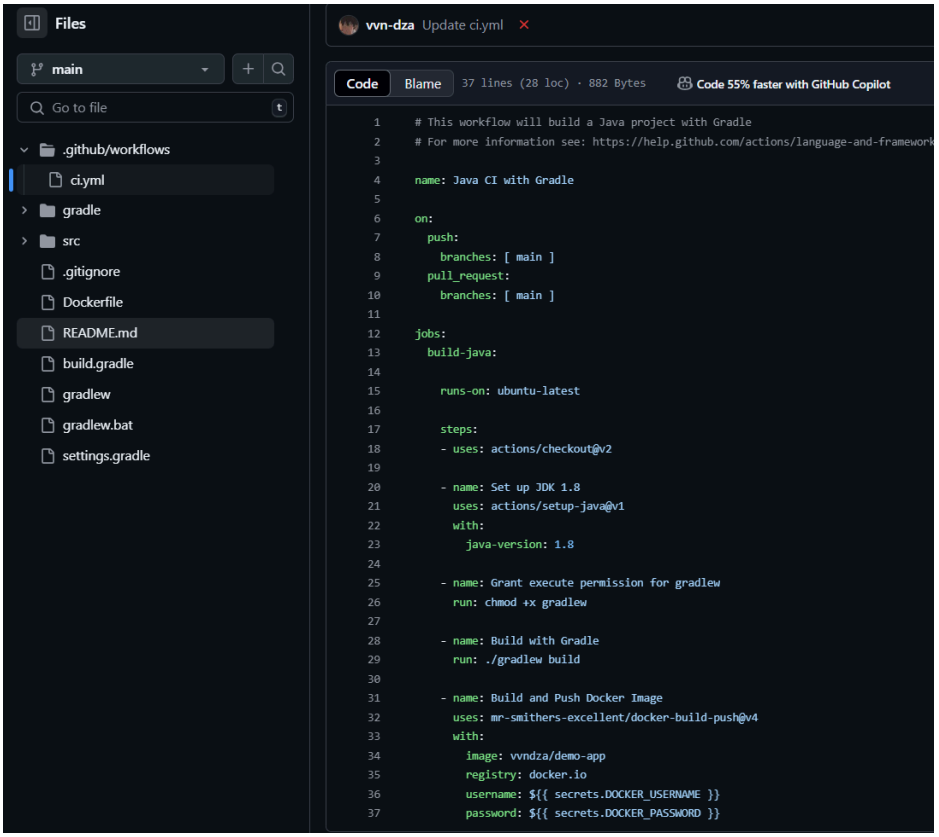
6.1 Syntax of Workflow File

The workflow file defines the automation pipeline for building and deploying the Java application.



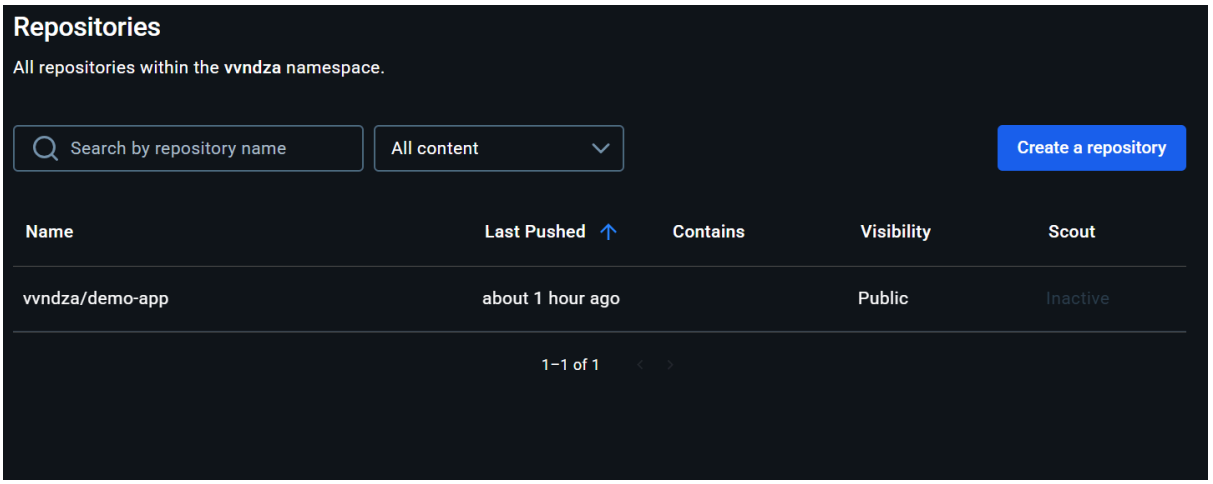
6.2 Where Does the Workflow Run?

GitHub Actions uses **GitHub Action Runners** to execute workflow steps in virtual environments.



6.3 Build Docker Image and Push to Private Docker Repo

Using Docker, we containerized the Java application and pushed it to a private Docker repository.



7. Java Gradle Repository and Docker Setup

We used a **Java Gradle** project from a GitHub repository and modified it to run in a Docker environment.

Dockerfile:

FROM openjdk:8-jre-alpine

EXPOSE 8080

COPY ./build/libs/my-app-1.0-SNAPSHOT.jar /usr/app/
WORKDIR /usr/app

ENTRYPOINT ["java", "-jar", "my-app-1.0-SNAPSHOT.jar"]

---

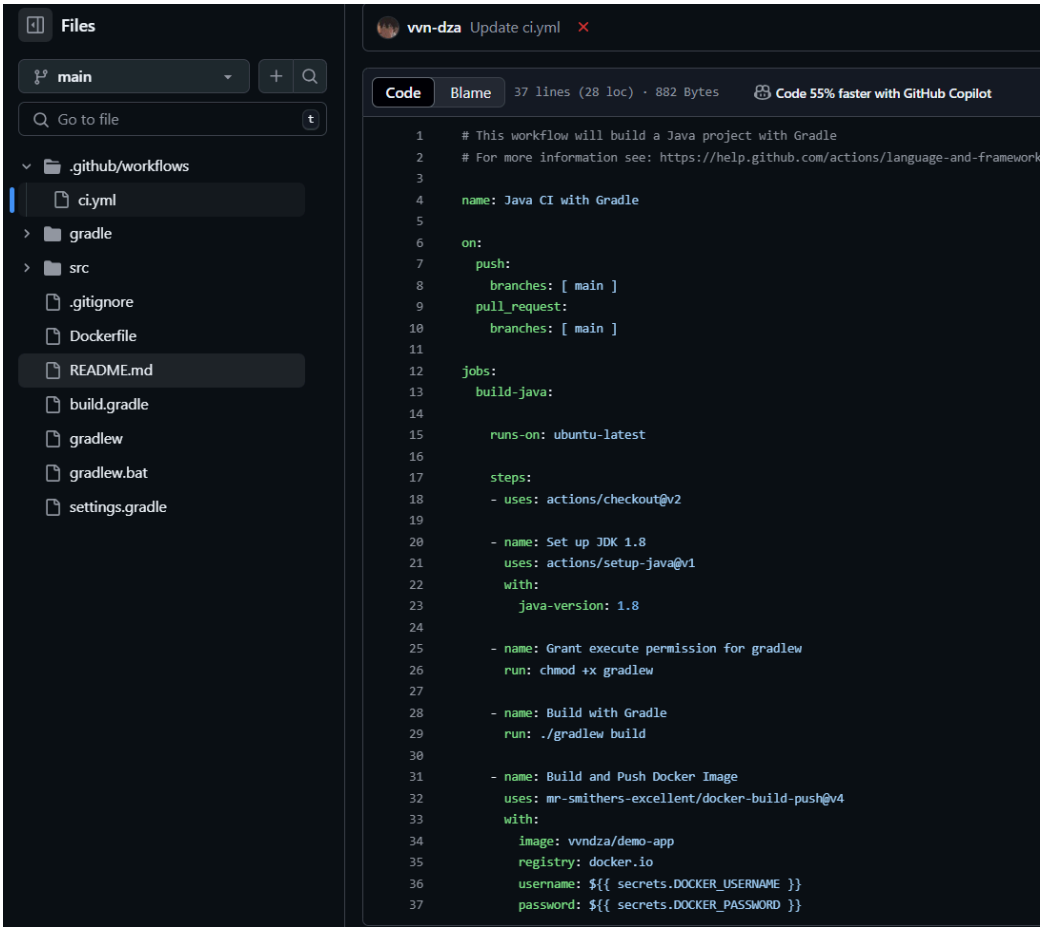
8. Workflow Action File

Below is the **GitHub Actions Workflow** file for automating the build and deployment process:

name: Java CI with Gradle

on:
  push:
    branches: [main]
  pull_request:

```
    branches: [ main ]

jobs:
  build-java:

    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v2

    - name: Set up JDK 1.8
      uses: actions/setup-java@v1
      with:
        java-version: 1.8

    - name: Grant execute permission for gradlew
      run: chmod +x gradlew

    - name: Build with Gradle
      run: ./gradlew build

    - name: Build and Push Docker Image
      uses: mr-smithers-excellent/docker-build-push@v4
      with:
        image: vvndza/demo-app
        registry: docker.io
        username: ${{ secrets.DOCKER_USERNAME }}
        password: ${{ secrets.DOCKER_PASSWORD }}
```



Conclusion

This document provides an overview of setting up a **GitHub Actions CI/CD Pipeline** for a Java application using **Gradle** and **Docker**. The workflow automates building, testing, and deploying the application efficiently.
