Steps to run CI and CD pipeline

Steps to Setup CI

1. Run SonarQube

Start a SonarQube instance using Docker:

```
docker run --name sonarqube-custom -p 9000:9000 sonarqube:10.6-community
```

- Open a browser and navigate to `http://localhost:9000`
- Use the default credentials `admin/admin`.
- You'll be prompted to change the password.
- Click **Create a new project** on the home page.
- Set the following details:
  - **Project Display Name**: `todoapp`
  - **Project Key**: `todoapp`
  - **Branch Name**: `main`
- Click **Next**.
- Select **Set up a project for Clean as You Code**.
- Select **Use global settings**.
- Click **Next**.
- Generate a token and copy it.
- Add this token to GitHub Secrets ,Set `SONAR_TOKEN` to `Copied token` in Github Secrets.
- Set `SONAR_HOST_URL` to `http://127.0.0.1:9000` in GitHub Secrets.

2. Run Self-Hosted

- Open `Windows Subsystem for Linux 2` (WSL2)
- Go to the `Settings` tab of your `repository` on `GitHub`.
- Navigate to the "`Actions`" section and Select `Runners`.
- Select "`Add self-hosted runner`" to initiate the process of adding a self-hosted runner.
- Choose the appropriate options:
  - **Operating System**: Select `Linux`.

    - **Architecture**: Select x64 (assuming your system architecture is 64-bit).

- After selecting `Linux`, follow the steps to set up the `self-hosted` runner.

- After configuration, the self-hosted runner should be operational and ready to execute workflows for your repository. You can use it in your workflow YAML files by specifying `runs-on: self-hosted`.

3. Configure Docker

- Add docker `DOCKER_USERNAME` to secrets
- Add docker `DOCKER_PASSWORD` to secrets

```
name: Continuous Integration

on:
  # push:
  #   branches:
  #     - main
  workflow_dispatch:

jobs:
  checkout:
    runs-on: self-hosted
    steps:
      - uses: actions/checkout@v4
        with:
          fetch-depth: 0  # Shallow clones should be disabled for a better
relevancy of analysis

  cache:
    runs-on: self-hosted
    needs: checkout
    steps:
      - name: Cache SonarQube packages
        uses: actions/cache@v4.0.2
        with:
          path: ~/.sonar/cache
          key: ${{ runner.os }}-sonar
          restore-keys: ${{ runner.os }}-sonar

      - name: Cache Maven packages
        uses: actions/cache@v4.0.2
        with:
          path: ~/.m2
          key: ${{ runner.os }}-m2-${{ hashFiles('**/pom.xml') }}
          restore-keys: ${{ runner.os }}-m2

  jacoco:
    runs-on: self-hosted
    needs: cache
    steps:
      - name: Build with Maven
        run: mvn -B package -Pcoverage
      - name: Generate JaCoCo badge
        id: jacoco
        uses: cicirello/jacoco-badge-generator@v2
        with:
          badges-directory: badges
          generate-branches-badge: true
          generate-summary: true

      - name: Log coverage percentages to workflow output
        run: |
          echo "coverage = ${{ steps.jacoco.outputs.coverage }}"
          echo "branches = ${{ steps.jacoco.outputs.branches }}"

      - name: Upload JaCoCo coverage report as a workflow artifact
        uses: actions/upload-artifact@v4.4.0
        with:
          name: jacoco-report
          path: target/site/jacoco/

  sonarqube:
    runs-on: self-hosted
    needs: jacoco
    steps:
      - name: Set up JDK 17
        uses: actions/setup-java@v1
        with:
          java-version: 17

      - name: Install Maven
        run: |
          if ! command -v mvn &> /dev/null; then
            echo "Maven not found. Installing Maven..."
            sudo apt update
            sudo apt install -y maven
          else
            echo "Maven is already installed."
          fi

      - name: Build and analyze
        env:
          SONAR_TOKEN: ${{ secrets.SONAR_TOKEN }}
          SONAR_HOST_URL: ${{ secrets.SONAR_HOST_URL }}
        run: mvn -B verify org.sonarsource.scanner.maven:sonar-maven-plugin:sonar
-Dsonar.projectKey=todoapp -Dsonar.projectName='todoapp'

  artifacts:
    runs-on: self-hosted
    needs: sonarqube
    steps:
      - run: mkdir staging && cp target/*.jar staging
      - uses: actions/upload-artifact@v4
        with:
          name: Package
          path: staging
  release:
    runs-on: ubuntu-latest
    needs: artifacts
    steps:
      - name: Checkout repository
        uses: actions/checkout@v4

      - name: Download web-app content
        uses: actions/download-artifact@v4.1.8
        with:
          name: Package

      - name: View content
        run: ls -R

      - name: Archive site content
        uses: thedoctor0/zip-release@master
        with:
          filename: app.zip

      - name: Create GitHub release
        id: create-new-release
        uses: actions/create-release@v1
        env:
          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
        with:
          tag_name: ${{ github.ref_type }}
          release_name: Release ${{ github.ref_type }}

      - name: Upload release asset
        uses: actions/upload-release-asset@v1
        env:
          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
        with:
          upload_url: ${{ steps.create-new-release.outputs.upload_url }}
          asset_path: ./app.zip
          asset_name: app-v${{ github.ref_type }}.zip
          asset_content_type: application/zip

  docker:
    runs-on: self-hosted
    needs: sonarqube
    steps:
      - name: Build Docker image
        run: docker build -t vijaynvb/todoapp .

      - name: Log in to Docker Hub
        run: echo "${{ secrets.DOCKER_PASSWORD }}" | docker login -u "${{
secrets.DOCKER_USERNAME }}" --password-stdin

      - name: Push image to Docker Hub
        run: docker push vijaynvb/todoapp
```

Steps to Setup CD

- Log in to your Google Cloud Console.
- In the left-hand menu, go to `IAM & Admin` > `Service Accounts`.
- Create a service account named `githubactions`.
- Assign the following roles to the service account:
    - `Kubernetes Engine Admin`
    - `Kubernetes Cluster Admin`
    - `Editor`
- Once the service account is created, you'll see it listed on the Service Accounts page
- Click on the Actions (three vertical dots) next to your service account and select Manage keys.
- Click `Add Key` > `Create New Key`.
- Select `JSON` as the key type and click Create
- Download the `JSON` key for this service account.
- Add the following secrets to your GitHub repository:
    - `GCP_PROJECT_ID`: Your Google Cloud `project ID`(find project id in google cloud console)
    - `GCP_SA_KEY`: The `JSON` key you downloaded
    - `GCP_SERVICE_ACCOUNT`: The email address of the service account

main.tf

```
provider "google" {
  project = var.project_id
  region  = var.region
}

resource "google_container_cluster" "primary" {
  name     = var.cluster_name
  location = var.location

  initial_node_count = 3

  node_config {
    machine_type = "e2-medium"
    oauth_scopes = [
      "https://www.googleapis.com/auth/cloud-platform",
    ]
    service_account = var.service_account
  }

  deletion_protection = false
}

variable "project_id" {
  description = "The ID of the GCP project"
}

variable "region" {
  description = "The GCP region"
}

variable "location" {
  description = "The GCP zone or location"
}

variable "cluster_name" {
  description = "The name of the GKE cluster"
}

variable "service_account" {
  description = "The service account for the GKE cluster"
}
```

deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: todoapih2
spec:
  replicas: 3
  selector:
    matchLabels:
      app: todo
  template:
    metadata:
      labels:
        app: todo
    spec:
      containers:
      - name: todoapih2
        image: vijaynvb/todoapp
        ports:
        - containerPort: 8081


---

apiVersion: v1
kind: Service
metadata:
  name: svctodoh2api
spec:
  selector:
    app: todo
  ports:
    - port: 80
      targetPort: 8081
  type: LoadBalancer
```

cd.yaml

```yaml
name: Continuous Deployment

on:
  workflow_run:
    workflows: ["Continuous Integration"]
    types:
      - completed
  workflow_dispatch:

env:
  CREDENTIALS: ${{ secrets.GCP_SA_KEY }}
  PROJECT_ID: ${{ secrets.GCP_PROJECT_ID }}
  GKE_CLUSTER: gke-todoapp-cluster   # cluster name
  GKE_REGION: us-east1
  GKE_LOCATION: us-east1-b      # cluster location
  SERVICE_ACCOUNT: ${{ secrets.GCP_SERVICE_ACCOUNT }}
  TFSTATE_CACHE_KEY: terraform-state-${{ github.sha }}

jobs:
  create-cluster:
    if: ${{ github.event.workflow_run.conclusion == 'success' }}
    runs-on: ubuntu-latest
    steps:
      - name: Checkout repository
        uses: actions/checkout@v4

      - name: Setup Terraform
        uses: hashicorp/setup-terraform@v1

      - name: Terraform Init
        run: terraform init
        env:
          GOOGLE_CREDENTIALS: ${{ env.CREDENTIALS }}

      - name: Terraform Plan
        run: terraform plan -input=false
        env:
          GOOGLE_CREDENTIALS: ${{ env.CREDENTIALS }}
          TF_VAR_project_id: ${{ env.PROJECT_ID }}
          TF_VAR_region: ${{ env.GKE_REGION }}
          TF_VAR_location: ${{ env.GKE_LOCATION }}
          TF_VAR_cluster_name: ${{ env.GKE_CLUSTER }}
          TF_VAR_service_account: ${{ env.SERVICE_ACCOUNT }}
      - name: Terraform Apply
        id: terraform-apply
        run: terraform apply -auto-approve -input=false
        env:
          GOOGLE_CREDENTIALS: ${{ env.CREDENTIALS }}
          TF_VAR_project_id: ${{ env.PROJECT_ID }}
          TF_VAR_region: ${{ env.GKE_REGION }}
          TF_VAR_location: ${{ env.GKE_LOCATION }}
          TF_VAR_cluster_name: ${{ env.GKE_CLUSTER }}
          TF_VAR_service_account: ${{ env.SERVICE_ACCOUNT }}

      - name: Cache Terraform State
        uses: actions/cache@v3
        with:
          path: |
            .terraform/
            terraform.tfstate
            terraform.tfstate.backup
          key: ${{ env.TFSTATE_CACHE_KEY }}

  deploy-to-cluster:
    runs-on: ubuntu-latest
    needs: create-cluster
    outputs:
      baseurl: ${{ steps.extract-url.outputs.baseurl }}
    steps:
      - name: Checkout repository
        uses: actions/checkout@v4

      - name: Install Google Cloud SDK
        uses: 'google-github-actions/auth@v2'
        with:
          credentials_json: ${{ env.CREDENTIALS }}

      - name: Set up Cloud SDK
        uses: 'google-github-actions/setup-gcloud@v2'

      - name: Configure kubectl to use gke-gcloud-auth-plugin
        run: gcloud components install kubectl

      - name: Get GKE credentials
        run: gcloud container clusters get-credentials ${{ env.GKE_CLUSTER }} --
zone ${{ env.GKE_LOCATION }} --project ${{ secrets.GCP_PROJECT_ID }}

      - name: Deploy application to GKE
        run: kubectl apply -f deployment.yaml

      - name: Wait for the service to be ready
        run: |
          echo "Waiting for 60 seconds..."
          sleep 60
      - name: Extract baseurl from GKE service
        id: extract-url
        run: |
          EXTERNAL_IP=$(kubectl get svc svctodoh2api -o
jsonpath='{.status.loadBalancer.ingress[0].ip}')
          PORT=$(kubectl get svc svctodoh2api -o jsonpath='{.spec.ports[0].port}')
          echo "EXTERNAL_IP is: $EXTERNAL_IP"
          echo "PORT is: $PORT"
          echo "baseurl=http://$EXTERNAL_IP:$PORT" >> $GITHUB_OUTPUT

  run-postman-tests:
    runs-on: ubuntu-latest
    needs: deploy-to-cluster
    steps:
      - name: Checkout repository
        uses: actions/checkout@v4

      - name: Install Newman
        run: npm install -g newman

      - name: Run Postman Collection
        env:
          BASEURL: ${{ needs.deploy-to-cluster.outputs.baseurl }}
        run: |
          newman run postman_collection.json --env-var "base_url=${{ env.BASEURL
}}"

  destroy-cluster:
    runs-on: ubuntu-latest
    needs: [deploy-to-cluster, run-postman-tests]
    steps:
      - name: Checkout repository
        uses: actions/checkout@v4

      - name: Restore Terraform State Cache
        uses: actions/cache@v3
        with:
          path: |
            .terraform/
            terraform.tfstate
            terraform.tfstate.backup
          key: ${{ env.TFSTATE_CACHE_KEY }}

      - name: Setup Terraform
        uses: hashicorp/setup-terraform@v1

      - name: Terraform Init
        run: terraform init
        env:
          GOOGLE_CREDENTIALS: ${{ env.CREDENTIALS }}

      - name: Terraform Destroy
        run: terraform destroy -auto-approve -input=false
        env:
          GOOGLE_CREDENTIALS: ${{ env.CREDENTIALS }}
          TF_VAR_project_id: ${{ env.PROJECT_ID }}
```

```
            TF_VAR_region: ${{ env.GKE_REGION }}
            TF_VAR_location: ${{ env.GKE_LOCATION }}
            TF_VAR_cluster_name: ${{ env.GKE_CLUSTER }}
            TF_VAR_service_account: ${{ env.SERVICE_ACCOUNT }}
```