Integrating GitHub Actions with Docker

Integrating GitHub Actions with Docker allows you to automate Docker-related tasks within your CI/CD pipelines directly from your GitHub repository. This integration facilitates the seamless building, testing, and deployment of Dockerized applications. Here's a brief explanation of how it works:

- **Workflow Definition:** GitHub Actions workflows are defined in YAML files within the .github/workflows/ directory of your repository. These workflows define a series of steps to be executed whenever certain events occur, such as pushing code changes or manually triggering the workflow.

- **Docker Setup:** Within your GitHub Actions workflow, you can set up Docker by using available actions or running Docker commands directly. Actions such as docker/setup-buildx-action can be used to set up Docker Buildx, which provides extended capabilities for building Docker images.

- **Building Docker Images:** Once Docker is set up, you can use Docker commands to build Docker images from your application's source code. Typically, this involves running a docker build command with appropriate options and arguments to generate a Docker image based on a Dockerfile.

- **Pushing Images to Registry:** After building the Docker image, you may want to push it to a Docker registry such as Docker Hub or a private registry. GitHub Actions allows you to use Docker login commands to authenticate with the registry using credentials stored as secrets, and then push the built image using docker push.

- **Handling Secrets:** Secrets such as Docker Hub credentials or other sensitive information required for the Docker-related tasks can be securely stored in GitHub repository secrets. These secrets can then be accessed within your GitHub Actions workflow, allowing you to authenticate with Docker registries without exposing sensitive information in your code.

> NOTE: Adding Environment Variables
>
> To add environment variables to this workflow, you can use GitHub Secrets, which allow you to securely store sensitive information.
>
> - Navigate to your GitHub repository.
> - Go to the "Settings" tab.
> - In the left sidebar, click on "Secrets".
> - Click on the "New repository secret" button.
> - Enter the name and value of your environment variable.
> - Click on "Add secret" to save it.
>
> In the provided workflow, environment variables are used for Docker Hub authentication. The `DOCKER_USERNAME` and `DOCKER_PASSWORD` secrets are used to log in to Docker Hub.

Docker.yaml

```yaml
name: 10 - Docker

on:
  workflow_dispatch: # Allows triggering the workflow manually from GitHub UI

jobs:
  docker-build:
    runs-on: ubuntu-latest

    steps:
    - name: Checkout repository
      uses: actions/checkout@v2

    - name: Set up JDK
      uses: actions/setup-java@v2
      with:
        java-version: '11'
        distribution: 'adopt'
        cache: maven  # Caches Maven dependencies to speed up builds

    - name: Build with Maven
      run: mvn clean install  # Builds the Java application using Maven

    - name: Build Docker image
      run: docker build -t padmanabhan1/todoapp .  # Builds a Docker image named
'padmanabhan1/todoapp'

    - name: Log in to Docker Hub
      run: echo "${{ secrets.DOCKER_PASSWORD }}" | docker login -u "${{
secrets.DOCKER_USERNAME }}" --password-stdin
      # Logs in to Docker Hub using the provided Docker Hub username and password
stored as secrets

    - name: Push image to Docker Hub
      run: docker push padmanabhan1/todoapp  # Pushes the Docker image to Docker
Hub
```

This workflow performs the following steps:

- Checks out the code from the repository.
- Sets up JDK 11 and Maven for building the Java application.
- Builds the Java application using Maven.
- Builds a Docker image named padmanabhan1/todoapp from the Dockerfile in the repository.
- Logs in to Docker Hub using the Docker Hub username and password stored as GitHub secrets.
- Pushes the built Docker image to Docker Hub.

To use this workflow:

- Create a file named docker.yml in the `.github/workflows/` directory of your repository.
- Paste the above YAML content into the `docker.yml` file.

- Ensure you have Docker Hub credentials stored as secrets named `DOCKER_USERNAME` and `DOCKER_PASSWORD` in your GitHub repository's settings.

- Ensure you have Docker Hub credentials stored as secrets named `DOCKER_USERNAME` and `DOCKER_PASSWORD` in your GitHub repository's settings.