Scenario 3 – Automating Image Builds using GitHub CI/Jenkins

In this exercise, you are a DevOps engineer at a growing tech startup. Your team needs a reliable and automated process to ensure that Docker images are always up to date with the latest code changes. Instead of manually building and pushing Docker images, you will leverage GitHub Actions to automate this process, ensuring that every change pushed to the main branch results in a freshly built and tagged Docker image on Docker Hub.

To uniquely identify each Docker image based on the specific code commit it was built from, you will use the commit hash as the Docker image tag instead of the conventional latest tag. This approach provides clear traceability, making it easy to identify exactly which version of the code each Docker image corresponds to.

Requirements

You need to set up a GitHub Actions workflow that will automatically build and push Docker images to Docker Hub whenever changes are pushed to the main branch of your GitHub repository. The Docker images should be tagged using the commit hash of the corresponding code.

Detailed Requirements and Acceptance Criteria:

1. **Create a New GitHub Repository**:
    a. You must create a new GitHub repository under your own account.
    b. Push the code from the Docker Mastery GitHub Repository (Branch: ***main***) to this new repository.
    c. Ensure that the Dockerfile created in the "Building Multi-Stage Docker Images" exercise is included in this repository.
2. **Create a GitHub Actions Workflow**:
    a. The workflow must trigger on any push to the main branch.
    b. It should check out the repository code, build the Docker image using the provided Dockerfile, and then push the image to Docker Hub.
    c. The Docker image must be tagged with the Git commit hash instead of latest.

    Significance of Using Commit Hash as the Tag:
    Tagging Docker images with the commit hash ensures that each image is uniquely identifiable and directly traceable to a specific code version. This eliminates

ambiguity and potential conflicts that can arise when using the latest tag, which can change over time as new images are built.

3. **Configure Docker Hub Authentication**:
    a. You must securely store your Docker Hub credentials using GitHub Secrets.
    b. The workflow should use these secrets to authenticate with Docker Hub and push the image.
4. **Verify the Automated Build**:
    a. After setting up the workflow, push a code change to the main branch. (You can make a change in README.md by adding anything to simulate a code change)
    b. Verify that the workflow successfully builds and pushes the image to Docker Hub, tagged with the commit hash.
    c. Pull the image from Docker Hub using the commit hash and run it to confirm that it works as expected.
5. **Acceptance Criteria**:
    a. The GitHub repository must be correctly set up with the code from the specified branch.
    b. The GitHub Actions workflow file should be correctly configured and trigger on pushes to the main branch.
    c. Docker images must be successfully built and pushed to Docker Hub, tagged with the corresponding commit hash.
    d. The pushed image must be verified by pulling and running it from Docker Hub using the commit hash.

Resources

- GitHub Actions Documentation
- Docker Hub Documentation
- Project Repository: Visit the Docker Mastery GitHub Repository
  Branch: main

Hints

- **Create a New GitHub Repository**: Clone the **main** branch of the project repository and push it to a new GitHub repository under your account.
- **GitHub Actions Workflow**: Create a .github/workflows/docker-build.yml file in your repository with the necessary steps to build and push the Docker image.

- **Docker Hub Authentication**: Use GitHub Secrets to securely store your Docker Hub username and password.
- **Tagging with Commit Hash**: Modify the workflow to extract the commit hash and use it as the tag for the Docker image.
- **Use Predefined Actions**: Github has a great marketplace of pre-created actions. You can use these instead of writing all workflow code by yourself. Checkout Github Marketplace.
- **Triggering the Workflow**: Make a code change [Changing anything in README,md], commit it, and push it to the main branch to trigger the workflow.