Session 2 Lab: Building Docker Images with GitHub Actions

Prerequisite

1. To complete this lab, you must have already done the prior lab. If you have not already done so, complete it using the following link: Session 1 Lab: Creating DevOps Pipelines with GitHub Actions

Creating a Dockerfile

1. Go to https://github.com, sign in, and open the repository you created in the previous lab.
2. Create a new branch called `session-2`. Make sure to spell the branch exactly because the name will be used later in your code.
3. Click the **Add file** or + sign button, and create a new file called `Dockerfile`. Paste the following code into it.

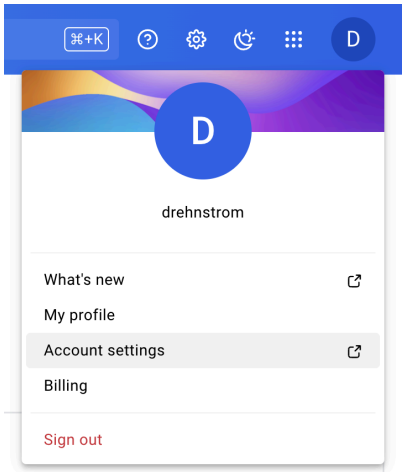
```Python
FROM python:3.11-slim

WORKDIR /app
COPY . .

RUN pip install gunicorn
RUN pip install -r requirements.txt

# Expose the port on which the Flask app will run
EXPOSE 8080

# Define the command to run the application using Gunicorn on port 8080
CMD ["gunicorn", "--bind", "0.0.0.0:8080", "app:app"]
```

Creating a Docker Hub Account and Credentials

1. Create a new browser tab, and go to **Docker Hub** using the following link: https://hub.docker.com/

2. If you already have a Docker account, sign in; otherwise, click the **Sign up** button to create a new one.

3. Click on your account icon in the upper-right corner and select **Account settings**.



4. From the Account settings page, select **Personal access tokens** from the **Security** section.

5. Click the **Generate new token** button and create a new access token named `GitHub Action Token`. Make sure the token is Read & Write. Paste the generated token in a text file; you will need it in a minute.

Running Actions with GitHub Secrets and Variables

1. Return to the browser tab with your GitHub repository. Click the **Settings** link.

2. From the Security section on the left, select **Secrets and variables**, and then **Actions**.

3. Click the **New repository secret** button. Create a secret called `DOCKER_HUB_USERNAME` with your Docker account name as the value. Then, create a second secret called `DOCKER_HUB_ACCESS_TOKEN` with the access token you just created.

4. Return to your GitHub repository **Code** view. Make sure you are using the `session-2` branch and navigate to the file `.github/workflows/run-tests.yml`. Open the file in **Edit** mode by clicking on the pencil icon. Replace the current contents of the file with the following code.

Python

```yaml
name: CI/CD Pipeline

on:
  push:
    branches:
      - session-1
      - session-2
  pull_request:
    branches:
      - main

jobs:
  test:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout code
        uses: actions/checkout@v3

      - name: Set up Python
        uses: actions/setup-python@v5
        with:
          python-version: '3.11'

      - name: Install dependencies
        run: |
          python -m pip install --upgrade pip
          pip install -r requirements.txt

      - name: Run tests
        run: |
          pytest --maxfail=1 --disable-warnings

  build_and_push:
    runs-on: ubuntu-latest
    needs: test
    steps:
      - name: Checkout code
        uses: actions/checkout@v3

      - name: Set up Docker Buildx
        uses: docker/setup-buildx-action@v2

      - name: Log in to Docker Hub
```

```
        uses: docker/login-action@v2
        with:
          username: ${{ secrets.DOCKER_HUB_USERNAME }}
          password: ${{ secrets.DOCKER_HUB_ACCESS_TOKEN }}

      - name: Build and push Docker image
        uses: docker/build-push-action@v5
        with:
          context: .
          push: true
          tags: ${{ secrets.DOCKER_HUB_USERNAME }}/tech-trek:${{ github.sha }}
ama
```

Note: This workflow now has two jobs: `test` and `build_and_push`. Take a look at how your secrets are being used in the second job.

There is also a variable `GitHub.sha` that is used to tag the image. This provides a unique name for every image created. Also note the parameter `needs: test` ensures that the `build_and_push` job is only run if the `test` job succeeds.

5. Commit the changes and then switch to the **Actions** tab. After a couple of seconds, your workflow should appear. When the workflow is complete, go back to https://hub.docker.com and refresh the page. You should have a new Docker image created.

Merge your Changes

1. Create a **Pull request** and merge the changes you made on the `session-2` branch with the `main` branch. Make sure to confirm and merge the pull request.
2. After you have merged the changes, go to the **Actions** menu. You should see that your pipeline ran when you did the pull request.