Lab 22: Automate CI/CD with GitHub Actions and Argo CD

Lab overview

In this lab, you will automate the container build and deployment process using GitHub Actions and Argo CD. You will configure and enable GitHub Actions to push container images to Docker Hub. You will also create a GitHub personal access token to allow your workflow to update your GitOps (DevOpsLab21) repository. Argo CD will then automatically detect changes to the repository and deploy the updated application to your Kubernetes cluster.

In this lab, you will:

- Create an access token on Docker Hub allowing you to read and write images
- Create a GitHub token to write to a repository
- Implement a GitHub Actions workflow to build, tag, and push Docker images upon code changes
- Update image tags to trigger application deployment with Argo CD
- Monitor automated deployment using Argo CD and verify visual changes in the application

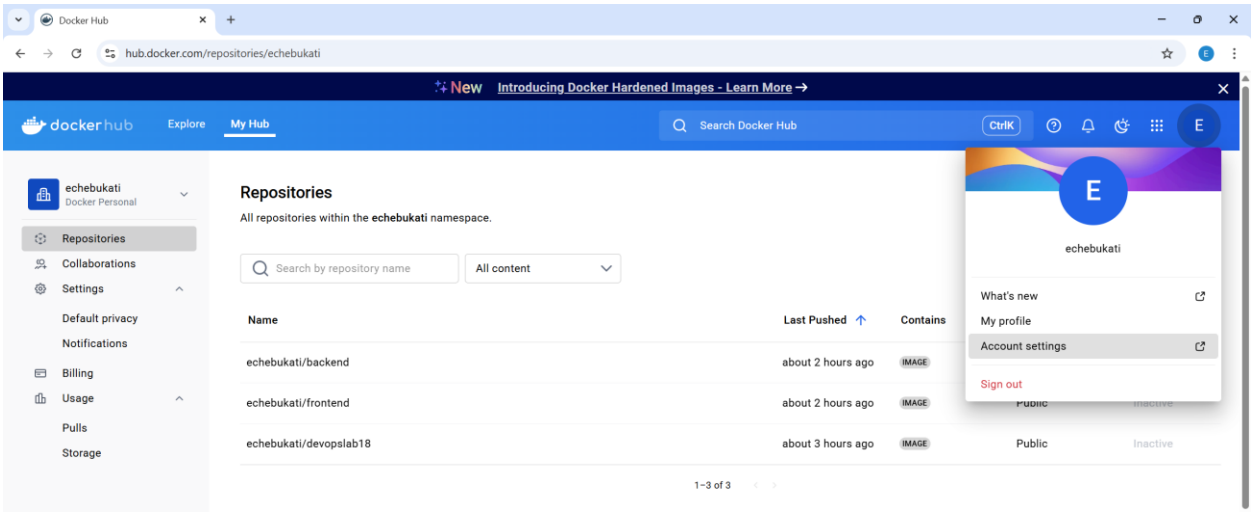
Estimated completion time

45 minutes

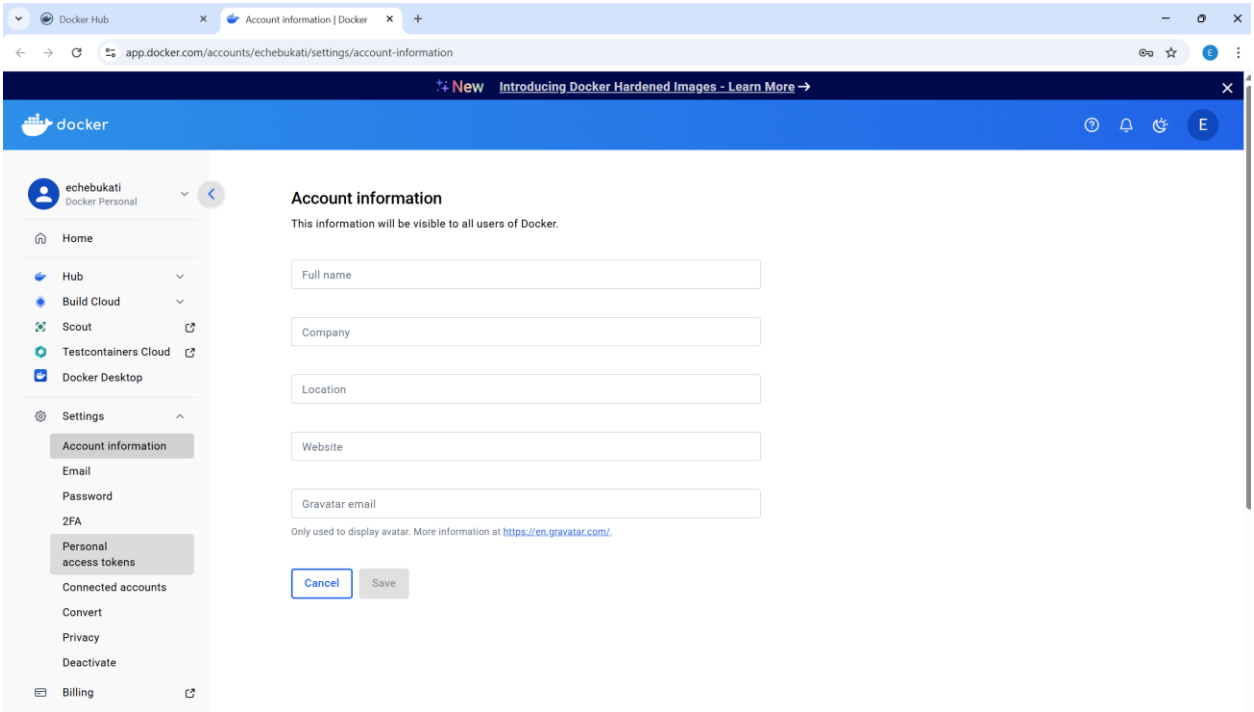
Task 1: Preparing your environment

In this task, you will create credentials that will allow you to push container images to Docker Hub. You will also create a GitHub token allowing you to push to the DevOpsLab21 repository from a GitHub Actions workflow in DevOpsLab20 repository.

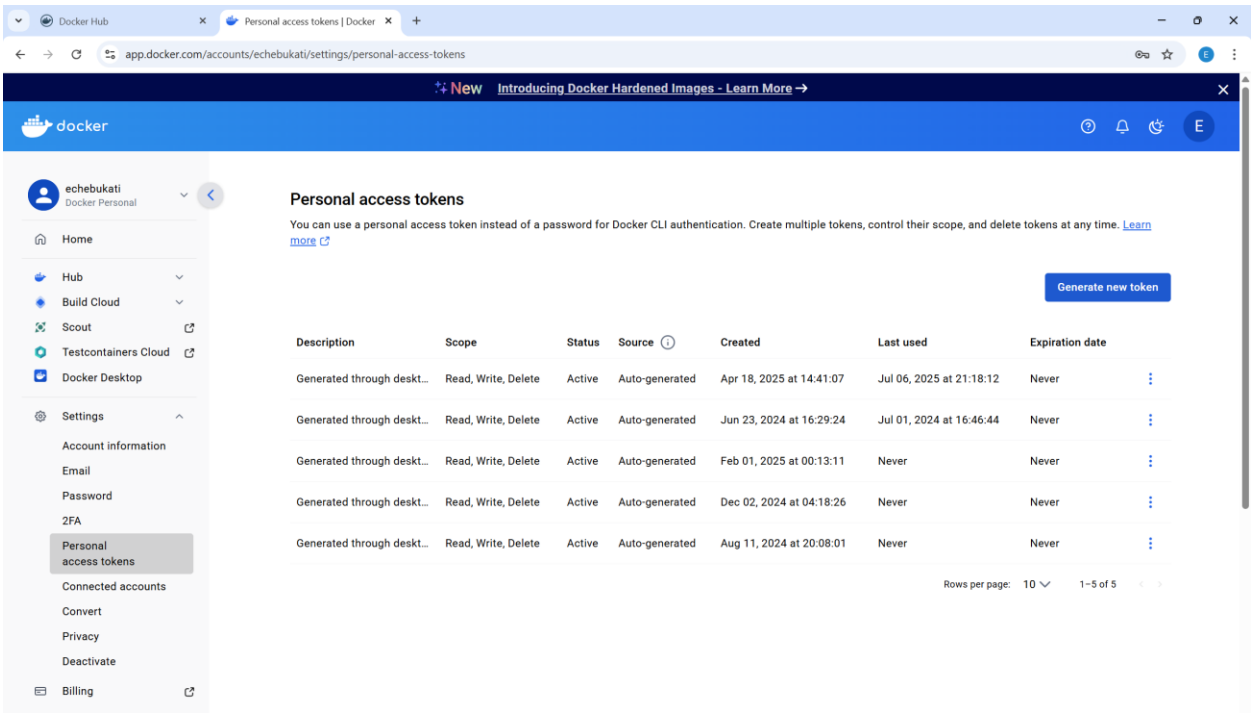
1. Open Docker Hub on your browser **https://hub.docker.com/repositories/**. Then, click on your avatar in the top-right corner and from the drop-down menu select **Account settings**.



2. Select **Settings** > **Personal access tokens**.



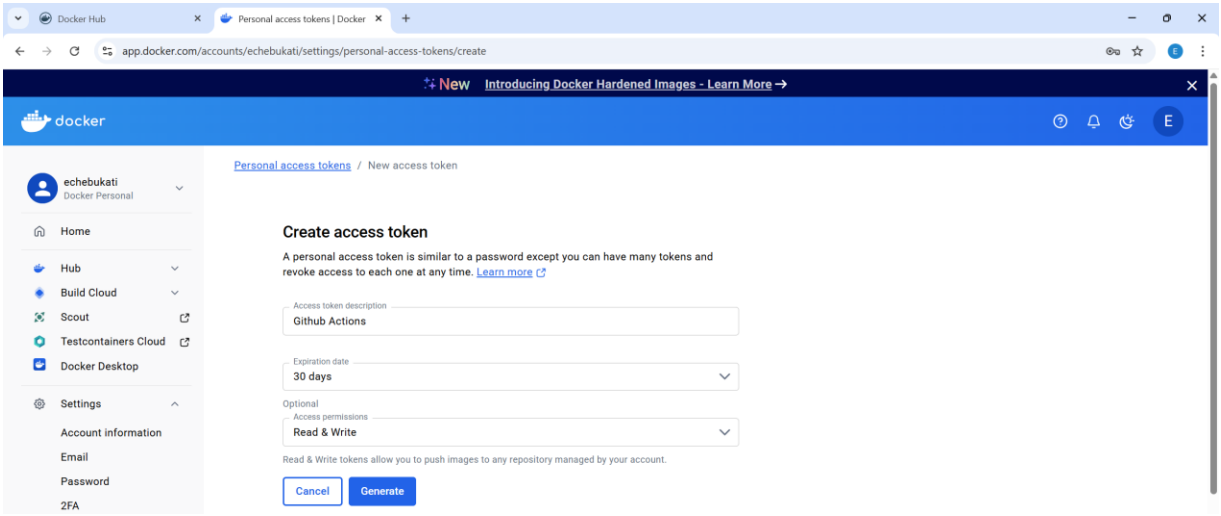
3. Click on **Generate new token**.



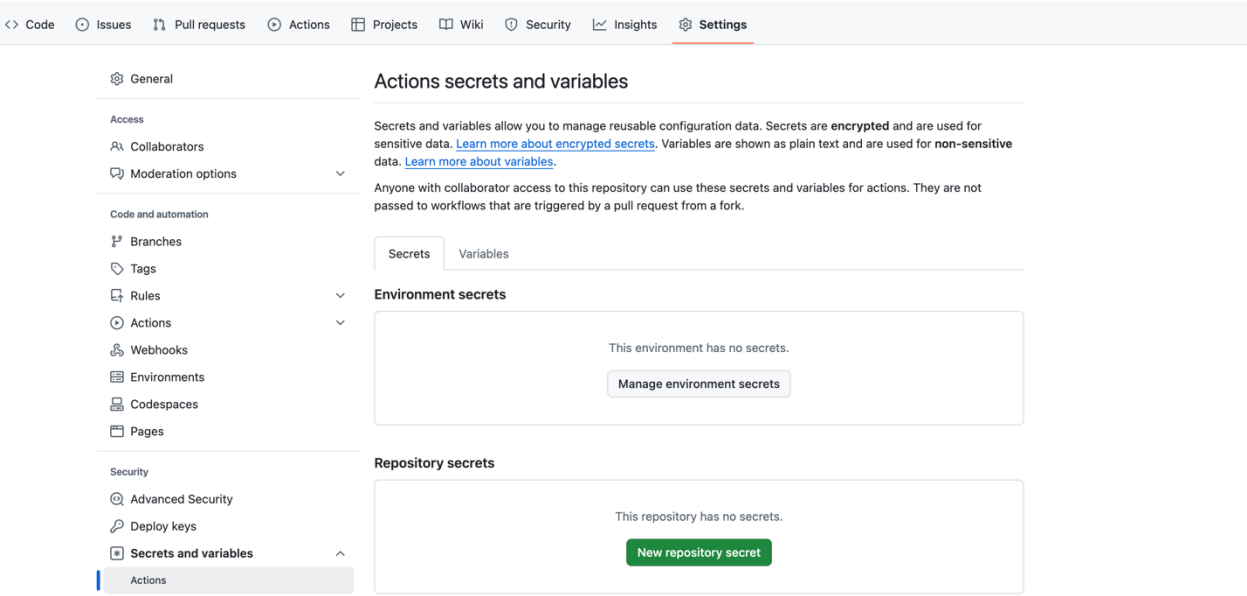
4. Scope the token as follows:
   - Access Token Description: **GitHub Actions**
   - Expiration: **30 Days**
   - Optional > Access permissions: **Read & Write**

5. Click **Generate**. Keep the access token safely.

> **Note**
>
> For the lab, you can save it into a notepad file - this is secure, as it will get destroyed when the labs are finished with.



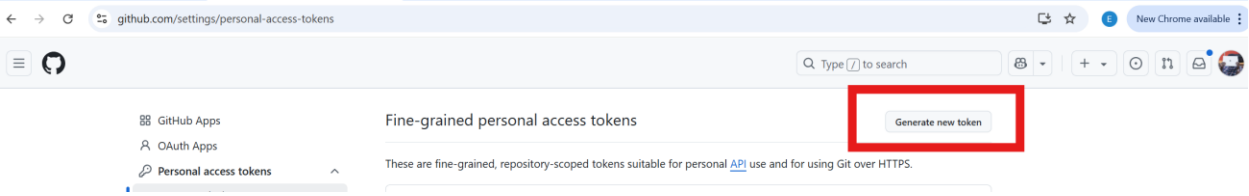
6. Go to your **DevOpsLab20** GitHub repository > **Settings** > **Secrets and variables** > **Actions**.



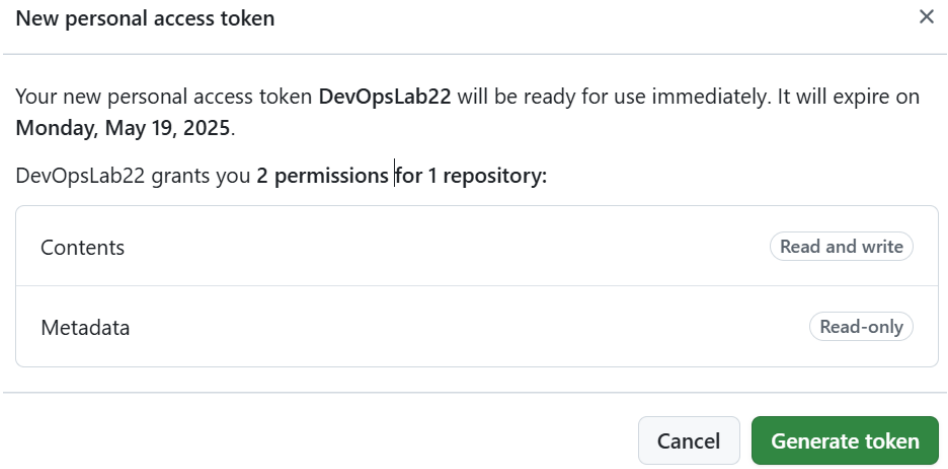
7. In the Repository secrets click on the **New Repository secret** button.
8. Add the following:

   DOCKER_PASSWORD Paste the full password acquired from Docker Hub into Secret pane.
9. While still on GitHub, you need to create a personal access token that will allow you to write to the DevOpsLab21 repository. Go to **https://github.com/settings/personal-access-tokens**.

10. Click **Generate new token**.



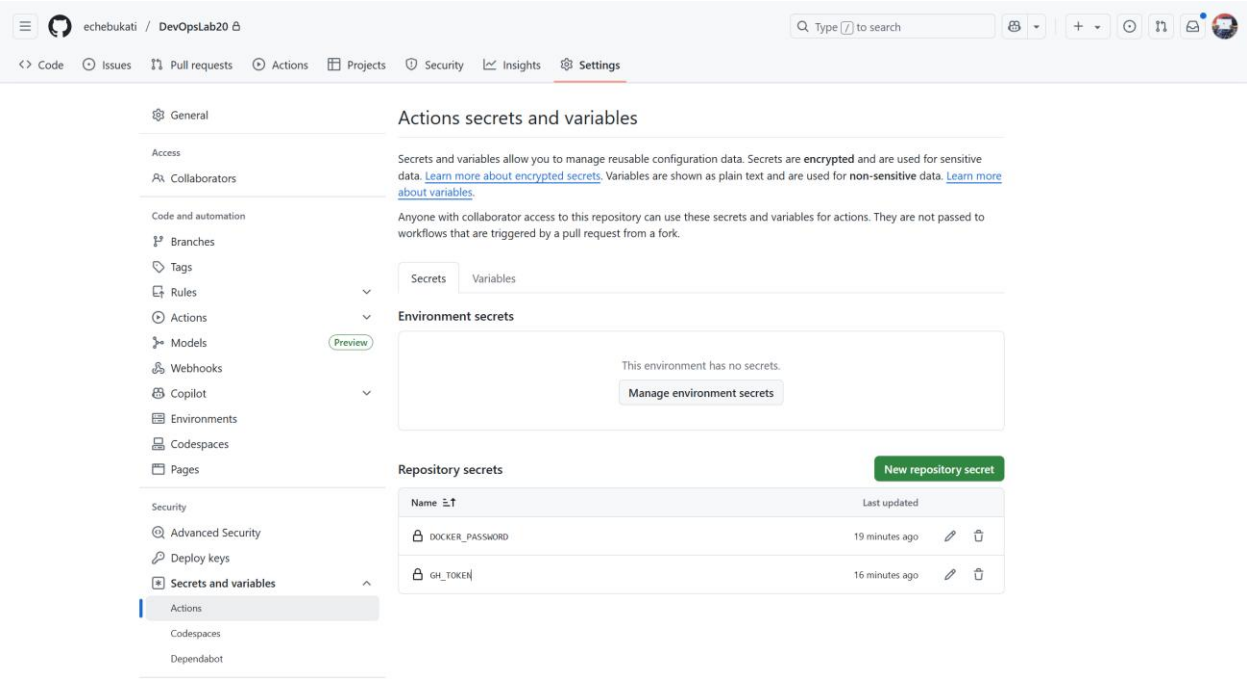
11. Scope the token as follows:
    - Token Name: **DevOpsLab22**
    - Expiration: **30 Days**
    - Repository Access: Only select repositories > **DevOpsLab21**
    - Permissions: Repository Permissions:-> Contents > **Read and Write**
12. Click **Generate token** and then confirm and **Generate token.**



13. Copy the personal access token and keep it safely.
14. Return to **DevOpsLab20** GitHub Repository > **Settings** > **Secrets and variables** > **Actions**.

15. Add the following:

    GH_TOKEN Paste the full token from step 12 into the details.



Task 2: Creating a GitHub Actions workflow in DevOpsLab20

In this task, you will create a GitHub Actions workflow in the **DevOpsLab20** GitHub repository which will be used to build and deploy the application code.

1. Create the folder **.github/workflows**, create a file called **build-and-deploy-frontend.yml** and input the following code. Replace ***<docker_username>*** with your Docker Hub username, and ***<your-github-username>*** with your GitHub username.

> **Note**
>
> A sample file is in the desktop folder Sample Lab Files/Lab22.

```
name: Build and Deploy Frontend


on:

  push:

    paths:
```

```
      - 'frontend/**'

jobs:
  build-and-deploy:
    runs-on: ubuntu-latest

    env:
      APP_NAME: frontend
      DOCKER_USERNAME: <docker_username>
      GITOPS_REPO: github.com/<your-github-username>/DevOpsLab21.git
      GITOPS_DIR: DevOpsLab21
      IMAGE_TAG: ${{ github.sha }}

    steps:
    - name: Checkout frontend source
      uses: actions/checkout@v4

    - name: Log in to Docker Hub
      uses: docker/login-action@v3
      with:
        username: ${{ env.DOCKER_USERNAME }}
        password: ${{ secrets.DOCKER_PASSWORD }}

    - name: Build and push Docker image
      run: |
        docker build -t $DOCKER_USERNAME/$APP_NAME:$IMAGE_TAG
./frontend
```

```
        docker push $DOCKER_USERNAME/$APP_NAME:$IMAGE_TAG


    - name: Clone GitOps repo

      run: git clone https://x-access-token:${{ secrets.GH_TOKEN
}}@$GITOPS_REPO


    - name: Update image tag in kustomization.yaml

      working-directory: ${{ env.GITOPS_DIR }}

      run: |

        sed -i "/name: $DOCKER_USERNAME\/$APP_NAME/{n;s|newTag:
.*|newTag: $IMAGE_TAG|}" apps/devopslab21/kustomization.yaml


    - name: Commit and push update

      working-directory: ${{ env.GITOPS_DIR }}

      run: |

        git config user.name "GitHub Actions Bot"

        git config user.email "ci@github.com"

        git add apps/devopslab21/kustomization.yaml

        git commit -m "chore($APP_NAME): update image to $IMAGE_TAG"

        git push https://x-access-token:${{ secrets.GH_TOKEN
}}@$GITOPS_REPO HEAD:main
```

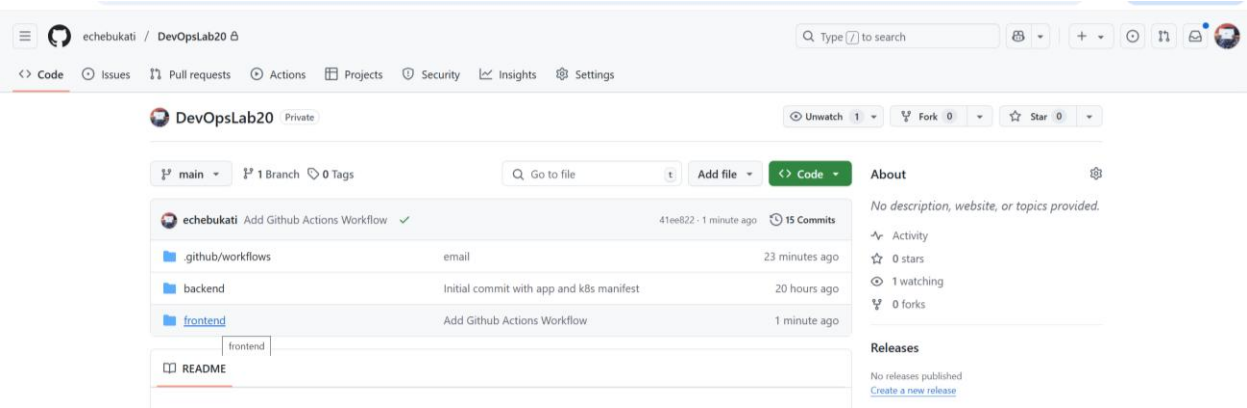
2. Commit and push these changes to GitHub. From the DevOpsLab20 folder:

```
git add .github/workflows/build-and-deploy-frontend.yml

git commit -m "Add Github Actions Workflow"

git push
```

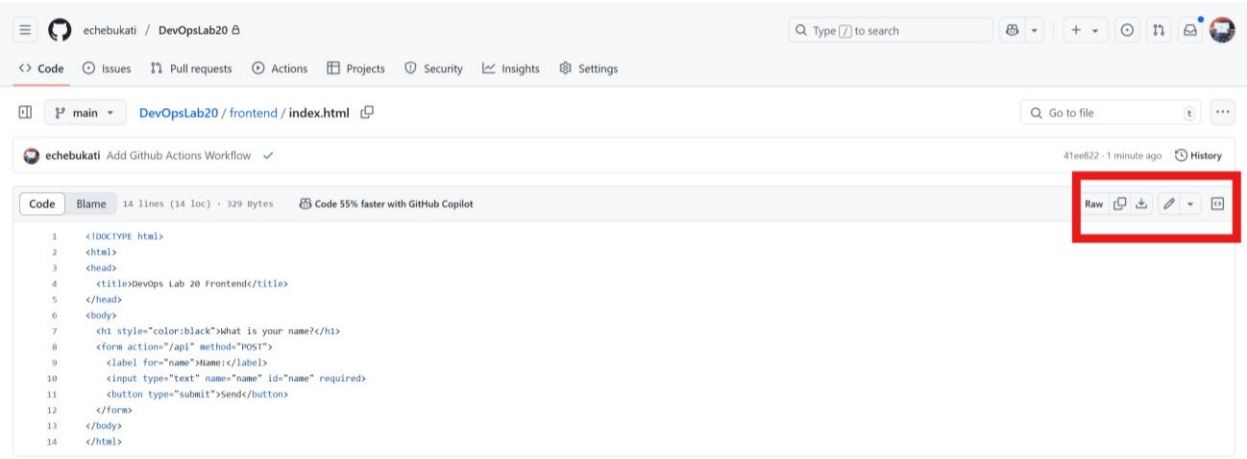
Task 3: Making changes to the application

In this task, you will make changes to your application, watch the GitOps repository (DevOpsLab21) have its image number changed. Then, you will monitor Argo CD as it deploys the application and view the changes on the browser.

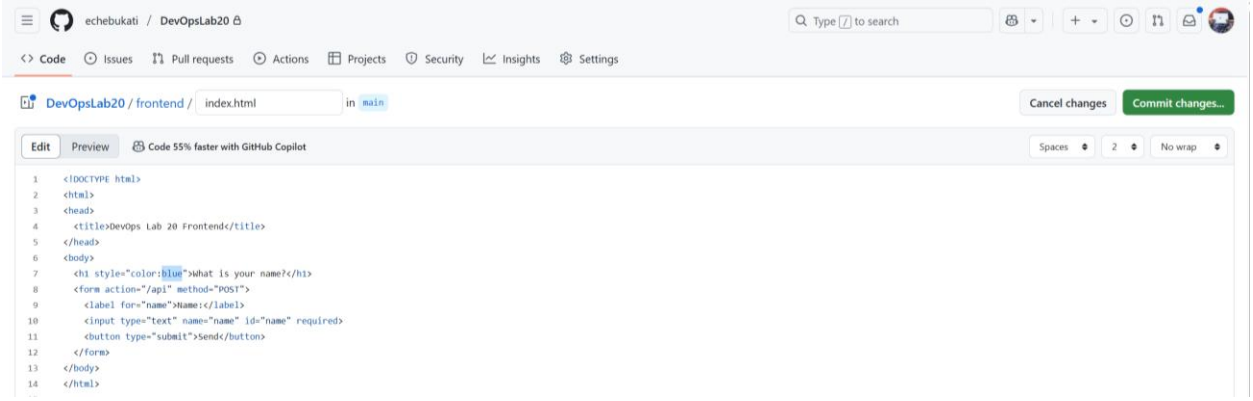
1. Open your DevOpsLab20 repository on GitHub. Click on the **frontend** folder.



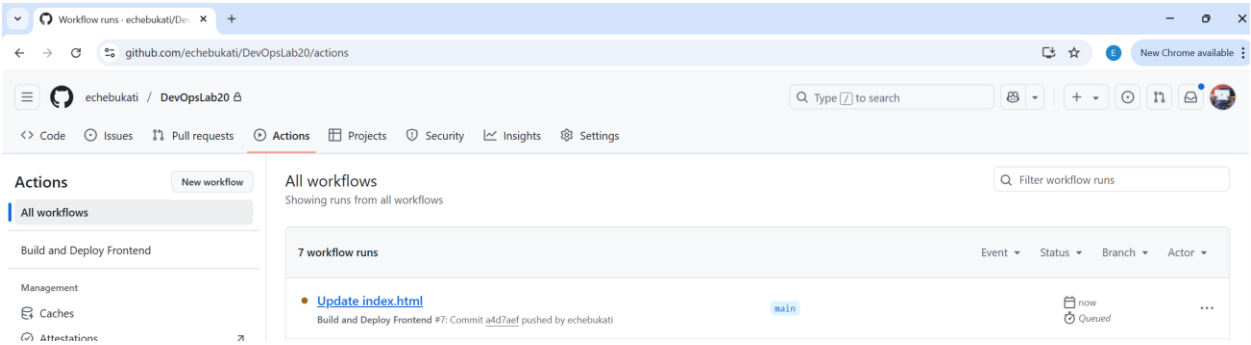
2. Click on **index.html** then click the **edit** icon.



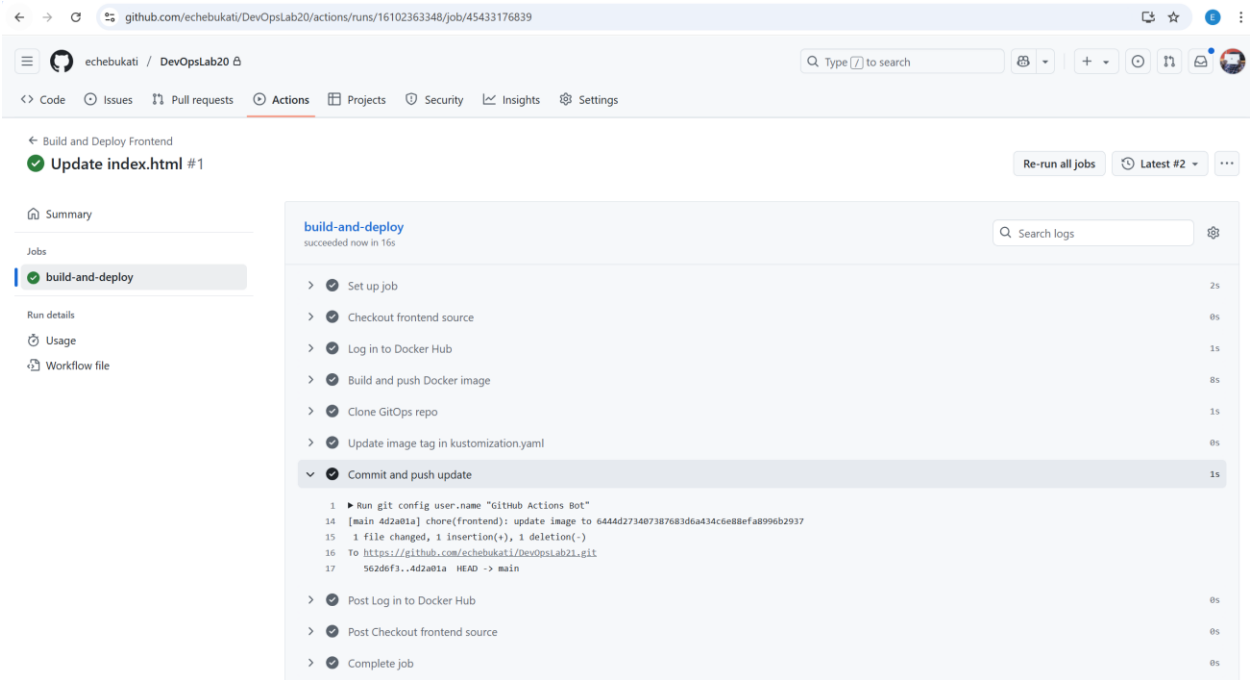
3. Modify the color of the heading from **black** to **blue** and commit changes directly to main.



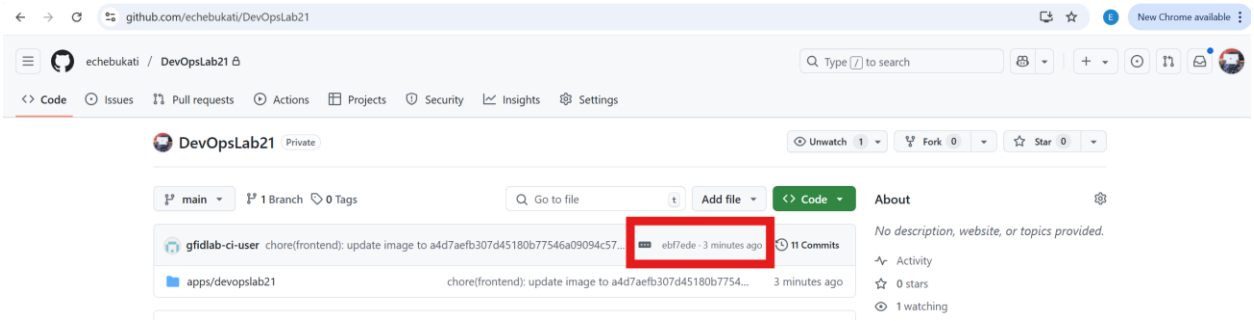
4. Go to the **Actions** tab and observe that a new workflow has been created. Click on it, then click on the **build-and-deploy** job.



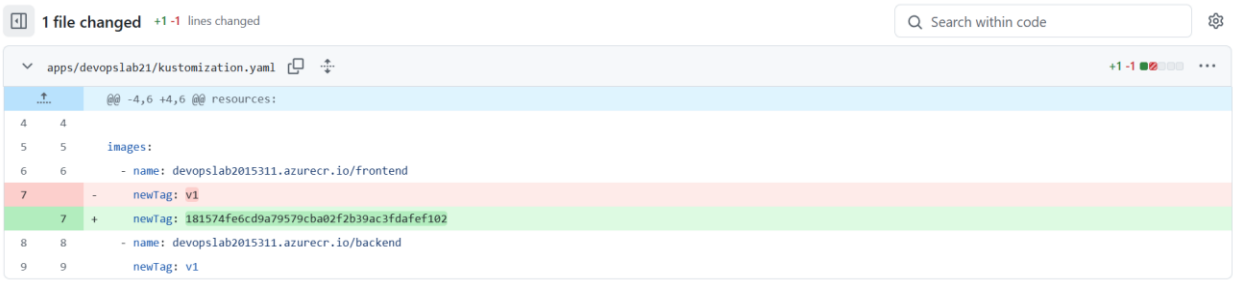
5. Wait for the workflow run successfully.



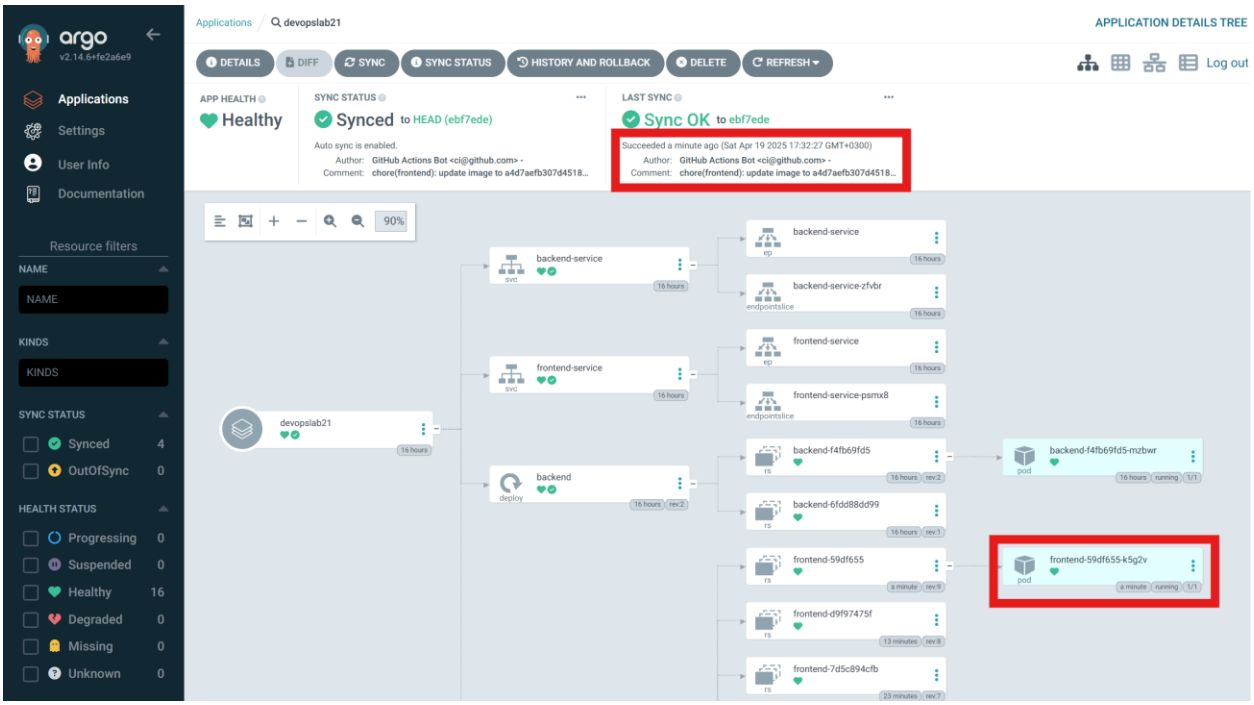
6. Go to your **DevOpsLab21** repository. Notice that there has been a recent commit. Click on it.



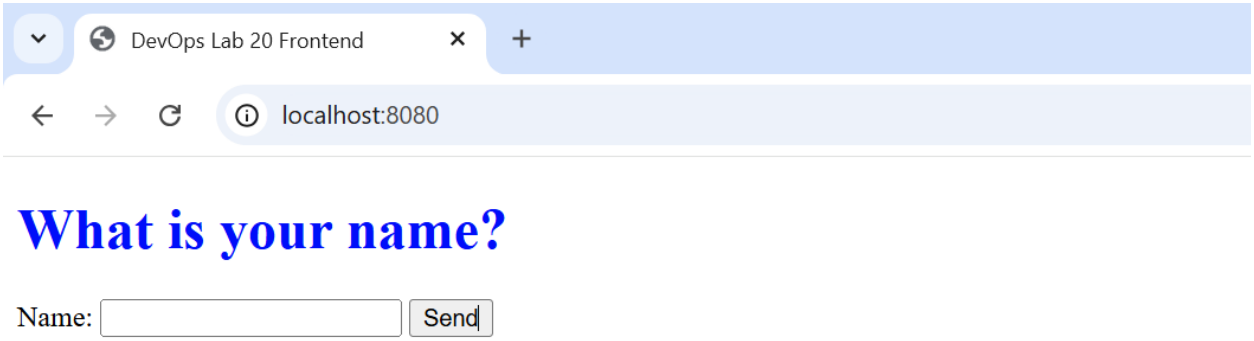
7. Note the change from the commit where only the image tag has changed.



8. Open up your Argo CD Dashboard from Lab 21 and navigate to your **devopslab21** application. Notice that a recent commit took place resulting in a NEW deployment of your frontend application pod.



9. Navigate to your frontend application on the browser and notice that the color of the Heading has changed from **black** to **blue**.



Lab review

1. What ultimately triggers the redeployment of the frontend application?
   A. A push to Docker Hub
   B. A push to the application repository (DevOpsLab20)
   C. A commit to the GitOps repository (DevOpsLab21)
   D. A commit to application repository (DevOpsLab20)
2. Why did the backend application not get updated with a new version?
   A. The backend application was disabled on Argo CD.
   B. There was no workflow created to update the image tag for the backend application.
   C. The backend application was not deployed using Argo CD.
   D. There were insufficient resources on GitHub Actions to deploy both applications.

STOP

You have successfully completed this lab.